MASTER EN SCIENCES INFORMATIQUES

Une extension du didacticiel Tcplp
adjonction du protocole ATM

Chavanne, Pierre

Award date:
1996

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année académique 1995 - 1996

Une extension du didacticiel Tcplp : adjonction du protocole ATM

Pierre Chavanne

Mémoire présenté en vue de l'obtention du grade de
Licencié et Maître en Informatique.

Résumé

Dans ce mémoire, nous abordons le protocole ATM sous l'angle d'un didacticiel de simulation. Nous nous sommes basés sur le didacticiel TcpIp pour y inclure une partie sur le protocole ATM. Nous présentons d'abord les raisons qui nous ont incité à développer un tel outil pour l'Enseignement Assisté par Ordinateur (E.A.O.). Ensuite, nous reprenons les idées d'une théorie sur la conception de didacticiel et nous faisons état des caractéristiques de base du protocole ATM. La partie centrale de ce mémoire reprend alors les spécifications et les détails de conception du didacticiel. Enfin, nous terminons par un passage en revue d'outils d'aide à la conception d'interfaces que nous aurions pu utiliser pour nous faciliter la conception de l'interface graphique.

Abstract

In this master thesis, we approach the ATM protocol through the presentation of a simulation software. We based our work on the education software TcpIp to include a part about the ATM protocol. First we introduce some reasons why to develop such C.A.T. software. We then lay out ideas out of a theory on C.A.T. software design and we make state of basic features of the ATM protocol. The main part of this master thesis includes the specifications and design details of our C.A.T. software. Finally, we review user interface software tools we could have used to help us designing our graphical interface.

Avant-propos

Je tiens tout d'abord à remercier mon promoteur, Philippe van Bastelaer, pour le temps qu'il a consacré à la lecture de ce mémoire et l'intérêt qu'il a porté à mon travail durant toute cette année.

Je souhaite également exprimer mes plus chaleureux remerciements à toute l'équipe du CEGELY de l'Ecole Centrale de Lyon pour l'accueil qu'ils m'ont réservé et les conseils qu'ils m'ont apportés tout au long de mon stage. Je désire remercier plus particulièrement Daniel Muller, Luc Mariaux et Alain Nicolas pour leur aide et le temps qu'ils m'ont consacré.

Je voudrais aussi remercier spécialement Xavier Gobert et Véronique Nachtergaele pour leur disponibilité et leurs conseils.

Je tiens finalement à remercier ma famille pour leur soutien tout au long de mes études et spécialement à l'occasion de la rédaction de ce mémoire.

Table des matières

Introduction

ATM est un terme que l'on retrouve de plus en plus fréquemment dans la littérature. Derrière ce nom se cache un protocole relativement jeune. Toutes les spécifications de ce protocole ne sont d'ailleurs pas encore définies par l'ATM Forum qui est chargé de sa standardisation. Cependant, les premiers réseaux ATM commencent à se répandre au fur et à mesure que les réseaux numériques à intégration de services, les fameux RNIS à bande large, font leur apparition. Il nous a donc semblé intéressant d'essayer de faire comprendre les concepts de base d'ATM par la réalisation d'un didacticiel.

Au cours de son stage lors de l'année académique précédente, Xavier Gobert avait réalisé un didacticiel dont le but était de faire comprendre le fonctionnement d'Internet et de certains types de réseaux qui le composent. Le nom TcpIp qui lui fut donné rappelle les noms des protocoles TCP et IP qui sont à la base d'Internet. C'est précisément ce didacticiel TcpIp qui va servir de base à notre projet. Nous avons donc travaillé sur TcpIp pour, dans un premier temps, en modifier quelque peu l'architecture pour la rendre plus conforme à la philosophie objet et, dans un deuxième temps, y ajouter les modules nécessaires au protocole ATM.

Le contexte dans lequel nous avons développé notre logiciel est décrit dans le premier chapitre. Nous expliquons les raisons qui nous ont poussés à développer un didacticiel sur ATM et nous faisons allusion au didacticiel TcpIp et à l'environnement COLOS dans lequel nous avons conçu notre produit.

Comme il s'agit du développement d'une partie d'un didacticiel, il nous a fallu suivre une méthodologie propre à ce genre de logiciel. La méthodologie dont nous nous sommes inspirés est décrite dans le second chapitre. Elle est tirée d'un ouvrage de référence en matière de conception de didacticiels. Cependant, cette méthodologie reste assez théorique. C'est pourquoi nous expliquerons, au quatrième chapitre, comment nous l'avons réellement appliquée lors de la phase de spécification de notre produit. Cette spécification porte sur la manière de représenter les concepts de base du protocole ATM que nous comptons expliquer. Ces concepts auront été présentés préalablement au cours du troisième chapitre.

Le cinquième chapitre détaille la conception du didacticiel. Il explique les nouvelles fonctionnalités, illustre les interfaces et décrit l'architecture logicielle. Le sixième et dernier chapitre quant à lui, donne un aperçu des types d'outils d'aide à la conception d'interfaces qui existent sur le marché. Nous mettons en évidence pour chacun de ces types d'outils, ses points faibles et ses avantages et nous évaluons son utilisation potentielle pour développer notre interface.

Chapitre 1

Un didacticiel pour ATM

Tout le monde entend de plus en plus souvent parler d'ATM. Mais quelle est réellement sa signification et sa portée aujourd'hui ? Nous allons dans ce premier chapitre décrire les raisons qui nous ont poussés à développer un logiciel de simulation pour le protocole ATM ainsi que le cadre dans lequel nous l'avons développé.

1.1. Un didacticiel de simulation du protocole ATM

1.1.1. La situation dans le domaine des télécommunications

Jusqu'à présent, la plupart des réseaux sont dédiés à des utilisations spécifiques telles que la téléphonie, la distribution de programmes de télévision, la commutation de circuits ou la transmission de données par paquets.

Certaines applications, comme la téléphonie, s'appuient sur le réseau téléphonique traditionnel. La mise en oeuvre de nouvelles applications sur des réseaux préexistants peut cependant provoquer des dysfonctionnements spécifiques, car ces infrastructures ne sont généralement pas conçues pour répondre aux besoins de services qui étaient inconnus au moment de leur conception. Ainsi, la transmission de données sur le réseau téléphonique est limitée à cause des possibilités réduites des équipements analogiques en matière de bande passante, de souplesse et de qualité. Les réseaux téléphoniques ont été conçus pour fournir un service à débit constant, alors que le support d'un trafic de données à vitesse variable nécessite des adaptations coûteuses.

Comme le réseau téléphonique était généralement incapable de supporter efficacement des services autres que la transmission de la voix, avec un service correspondant aux exigences de clients, des infrastructures spécifiques se sont développées. Parmi celles-ci, on trouve les

réseaux publics de transmission de données ou des installations privées. Internet constitue un exemple de réseau de transmission de données de taille importante.

Les réseaux privés reposent souvent sur des équipements, des interfaces et des protocoles spécifiques. En outre, ils sont inaccessibles depuis d'autres réseaux ou par des utilisateurs externes. Dans ce cas, il est nécessaire d'installer des passerelles permettant de communiquer avec le monde extérieur, ce qui peut constituer une opération lourde et coûteuse.

1.1.2. Le concept de RNIS

En 1984, l'assemblée plénière du CCITT a adopté les premiers avis qui définissent les caractéristiques d'un réseau numérique à intégration de service (RNIS). Le CCITT déclare que "un RNIS est un réseau (...) qui fournit une connexion numérique de bout en bout permettant de supporter un large éventail de services, vocaux et non-vocaux, auxquels les utilisateurs peuvent accéder à travers un ensemble limité d'interfaces usager-réseau polyvalentes et normalisées"[HAN95].

Le RNIS d'origine est basé sur le réseau téléphonique numérique. Les réseaux numériques à intégration de services sont en cours de déploiement depuis le début des années 90. Ce type de RNIS est appelé *RNIS à bande étroite*. Les avantages que ces réseaux numériques présentent pour l'utilisateur et pour l'opérateur de réseau sont les suivants :

- interface usager-réseau unique permettant l'accès à une grande variété de services,
- capacités évoluées de signalisation,
- intégration de services,
- fourniture de services nouveaux ou améliorés.

1.1.3. Le RNIS à large bande

Le débit maximum sur un RNIS à bande étroite est d'environ 1,5 à 2 Mbit/s. Mais l'interconnexion de réseaux locaux ou la transmission de séquences vidéo avec une résolution correcte nécessitent dans un grand nombre de cas des débits considérablement plus élevés. Par conséquent, le besoin de concevoir et de réaliser un RNIS à bande large s'est imposé. Les débits mis à la disposition des utilisateurs vont d'environ 50 Mbit/s à quelques Gbit/s.

Après de nombreuses discussions sur la manière de réaliser l'interface usager-réseau, la solution qui fut retenue est basée sur la division de la capacité utile en éléments de taille réduite appelés *cellules*, chacune pouvant servir à un usage quelconque. Chaque cellule peut être utilisée pour véhiculer des informations associées à n'importe quel type de connexion.

Ce principe a été implémenté dans le mode de transfert **ATM** (*Asynchronous Transfer Mode*).

Le développement du RNIS à large bande ne se justifie et ne connaîtra le succès qu'à condition qu'il réponde aux besoins des clients potentiels. En principe, ce RNIS devrait présenter un intérêt à la fois pour les entreprises et les particuliers. Il est donc nécessaire d'envisager la transmission de données mais également les possibilités liées à la distribution de programmes de télévision ou à d'autres activités de loisirs.

Le RNIS à large bande supportera des services mettant en oeuvre des débits constants ou variables, la transmission de données, de voix (sons), d'images fixes ou animées, ou, ce qui est particulièrement intéressant, des applications multimédia susceptibles de combiner données, voix et images.

1.1.4. Les premiers services disponibles sur ATM

La totalité des services envisageables sur le RNIS à large bande ne pourront être offerts dès le démarrage d'ATM pour plusieurs raisons. Tout d'abord la plupart d'entre eux ne sont pas complètement définis. Ensuite, il sera nécessaire d'identifier un sous-ensemble de ces services susceptibles d'intéresser des clients potentiels. Celui-ci devra inclure des applications existantes. Cependant il est important, pour des aspects marketing que les nouveaux apports spécifiques d'ATM soient visibles.

Les fonctions de base d'un réseau ATM sont le transport et l'acheminement de cellules ATM. Le réseau n'a pas à connaître quoi que ce soit concernant les applications qui utilisent une connexion ATM. Les usagers peuvent utiliser ce service ATM pour échanger sur le réseau, des données, de la voix, des images ou une combinaison des trois. Cela devrait pousser les utilisateurs à expérimenter de nouvelles applications sur ATM, telles que le multimédia.

Le besoin d'intégrer les services interactifs et de diffusion ainsi que les modes de transmission par circuits et par paquets dans un réseau à large bande universel est une des raisons qui sont à l'origine de l'incorporation de caractéristiques large bande dans le RNIS. Comparée à la juxtaposition de plusieurs réseaux spécialisés, l'intégration de ces réseaux et de ces services présente des avantages majeurs en ce qui concerne les prévisions économiques, le développement, l'implémentation, l'exploitation et la maintenance. Là où les réseaux dédiés obligent le client à disposer de plusieurs lignes d'accès distinctes et coûteuses, une seule fibre optique suffit pour accéder au RNIS à large bande. La production à grande échelle de composants d'un haut degré d'intégration pour un seul type de RNIS à large bande va permettre d'arriver à des solutions économiquement rentables [HAN95].

1.1.5. Pourquoi un logiciel sur ATM ?

Alors que la plupart des réseaux de télécommunication pré-RNIS ont été des réseaux spécialisés (par exemple pour le téléphone ou les données), offrant des débits limités et des capacités de traitement réduites, le futur RNIS à large bande est conçu comme un réseau (normalisé) universel supportant différents types d'applications et de catégories de clients.

Les implémentations du RNIS à large bande seront basées, selon l'ITU-T, sur le mode de transfert asynchrone (ATM). ATM est donc destiné à devenir le protocole du RNIS à large bande.

En raison de sa vocation à devenir un protocole universel regroupant les fonctionnalités de tous les protocoles existants, ATM est extrêmement complexe. Il doit prévoir un mécanisme pour simuler la commutation de circuits et la communication par paquets, de même que pour gérer tout le système de cellules et garantir tous les services souhaités par les utilisateurs. Ce protocole s'appuie, comme tous les protocoles réseaux, sur un modèle en couches. Cependant, la différence avec les autres protocoles réside dans le nombre de couches et les fonctionnalités

différentes associées aux couches du modèle ATM. En effet le modèle du RNIS à large bande qui inclut les couches du protocole ATM se distingue du modèle classique OSI (*Open System Interconnection*) et du modèle du DOD (*Department Of Defense*).

Pour ces raisons, et vu l'intérêt que nous portons à l'EAO (Enseignement Assisté par Ordinateur) et aux possibilités que ce mode d'enseignement apporte dans le domaine de l'informatique, nous avons choisi de réaliser un didacticiel permettant de simuler par une animation, les concepts de base régissant le fonctionnement de ce protocole ATM.

Bien que ce futur RNIS concerne l'ensemble de la population, le didacticiel que nous avons développé s'adresse à des personnes ayant déjà une certaine connaissance de base en matière de protocoles de réseaux informatiques. En effet, les futurs utilisateurs se verront davantage attirés par les possibilités offertes par ce nouveau réseau que par les aspects techniques propres au mode de transfert utilisé, ces détails techniques étant du ressort des informaticiens et autres ingénieurs spécialisés dans les réseaux informatiques.

1.2. Le didacticiel TcpIp.

Dans le domaine des télécommunications, parmi la liste des didacticiels existants, TcpIp fut développé en 1995 par Xavier Gobert lors de son stage effectué dans le cadre de son curriculum de licence et maîtrise en informatique [XGO95].

Ce logiciel sert de base au travail que nous avons réalisé. Nous allons d'abord décrire brièvement l'objet même de ce logiciel, c'est-à-dire Internet. Ensuite, nous résumerons les lignes directrices du logiciel TcpIp. Et enfin, nous parlerons de la manière dont notre projet fut intégré dans le produit existant en abordant les nouveautés introduites par rapport à la version initiale du produit.

1.2.1. Internet : un réseau de réseaux

Le didacticiel TcpIp est un logiciel de formation destiné à fournir un apprentissage des notions de base d'Internet. Internet est devenu un mot très à la mode ces derniers temps mais bien peu de gens dans le grand public connaissent la signification de ce concept.

I. Origines d'Internet

Internet a vu le jour dans les années 1970. Tel que nous le connaissons aujourd'hui, Internet est le résultat d'un programme de recherche mené aux Etats-Unis par le DOD (Department of Defense). La DARPA (Defense Advanced Research Projects Agency) a financé le développement d'un réseau expérimental *ARPAnet* qui fonctionnait sur le principe d'une commutation de paquets. Ce mode de commutation présentait comme avantages, outre une meilleure utilisation des ressources informatiques et une réduction des coûts d'utilisation, des possibilités de routage en cas de congestion du réseau ou de problème sur un itinéraire initial. Un des principaux objectifs poursuivis par le DOD était de développer une technologie de réseau qui pourrait résister à de sérieux dommages comme ceux engendrés par une attaque nucléaire.

La disponibilité de fonds pour la recherche fournis par la DARPA ne manqua pas d'attirer l'attention de plusieurs groupes de recherche. La DARPA planifia des réunions informelles pour que les chercheurs partagent leurs idées et discutent de leurs résultats. En 1979, le succès fut tel

que la DARPA constitua un comité pour coordonner et guider la conception de protocoles et d'architectures de l' "Internet commuté".

L'Internet débuta vers 1980 quand la DARPA commença à introduire les protocoles TCP/IP dans les machines connectées aux réseaux de recherche. L'ARPAnet est devenu l'épine dorsale de cet Internet et fut utilisé pour les premières expériences avec TCP/IP. La transition vers la "technologie Internet" s'est achevée en janvier 1983 quand le bureau du Secrétariat à la Défense américain obligea tout ordinateur connecté à un réseau à longue distance d'utiliser TCP/IP. Au même moment, le DCA (Defense Communication Agency) divisa l'ARPAnet en deux réseaux séparés : un pour la recherche et l'autre pour les communications militaires. La partie recherche a gardé le nom d'ARPAnet alors que la partie militaire fut appelée MILNET.

Pour encourager les chercheurs universitaires à adopter et utiliser les nouveaux protocoles, la DARPA créa une implémentation de ses protocoles à faible coût. A cette époque, la plupart des départements informatiques des universités utilisaient UNIX comme système d'exploitation. En finançant Bolt Beranek and Newman pour implémenter ses protocoles TCP/IP pour être utilisés sous UNIX et en finançant Berkeley pour qu'ils intègrent les protocoles dans les logiciels qu'ils fournissaient, la DARPA a réussi à toucher 90% des départements informatiques des universités. De plus, le nouveau logiciel utilisant les protocoles arriva sur le marché à un moment propice. En effet beaucoup de départements étaient en train d'acquérir leur second ou troisième ordinateur et voulaient les connecter entre eux avec un LAN. Ces départements avaient besoin de protocoles de communication et aucun autre n'était généralement disponible.

La distribution du logiciel de Berkeley s'est répandue parce qu'ils offraient plus que les protocoles TCP/IP de base. Berkeley offrait un ensemble d'utilitaires pour des services réseau qui ressemblaient aux services UNIX utilisés sur une machine unique. Le succès de la technologie TCP/IP et de l'Internet parmi la communauté scientifique amena d'autres groupes à l'adopter.

II. Les protocoles d'Internet

L'Internet n'est pas un réseau unique mais plutôt un réseau de réseaux. Tous ceux-ci respectent un ensemble commun de protocoles pour supporter les connexions et la gestion du trafic. Le couple de protocoles le plus utilisé est actuellement TCP/IP.

III. Les protocoles de "sous-réseaux"

Comme nous venons de le mentionner, Internet est un réseau de réseaux. Chacun de ces différents réseaux peut être vu comme un élément de l'Internet respectant le protocole défini pour être en communication avec les autres éléments d'Internet. Cependant, tous ces réseaux que l'on peut appeler sous-réseaux n'en constituent pas moins des réseaux indépendants à part entière. Ils utilisent donc chacun leur protocole de communication permettant aux machines en faisant partie de "*dialoguer*" entre elles.

1.2.2. Le didacticiel TcpIp

Le didacticiel TcpIp, comme son nom l'indique, porte sur les protocoles majeurs utilisés dans Internet. Le but poursuivi par ce logiciel est de simuler graphiquement une communication entre deux machines appartenant à Internet. Pour ce faire, la communication est décomposée au sein de chaque sous-réseau interconnecté via Internet. En effet, chaque machine utilise des primitives de communication propres au sous-réseau auquel elle appartient. La simulation présentée reste somme toute à un niveau logique assez élevé. Elle montre le déroulement d'une communication au niveau d'interconnexion réseau.

Les protocoles TCP/IP se situent au niveau interconnexion. Ils permettent donc à des applications de communiquer via des réseaux supportant des protocoles différents. Le didacticiel TcpIp est donc un cadre dans lequel peuvent venir s'insérer des modules aux niveaux inférieurs (sous-réseaux avec de nouveaux protocoles) et aux niveaux supérieurs (applications).

De plus, le nombre de types de sous-réseaux présents dans l'Internet et représentables initialement dans TcpIp est fixe et limité à cause de l'espace disponible sur un écran d'ordinateur. La configuration rigide d'Internet choisie est illustrée à la *Figure 1-1*.

Cette configuration est constituée de deux réseaux locaux (LAN), d'un réseau à commutation de circuits (CSPDN), d'un réseau à commutation par paquets (PSPDN) et d'une ligne louée. Chaque type de sous-réseau est représenté graphiquement avec une toile de fond différente. Une vue détaillée du déroulement d'une communication au sein d'un sous-réseau peut être demandée par l'utilisateur. Toutefois, cette vue se résume pratiquement à un agrandissement de ce qui se passe au niveau de l'Internet.

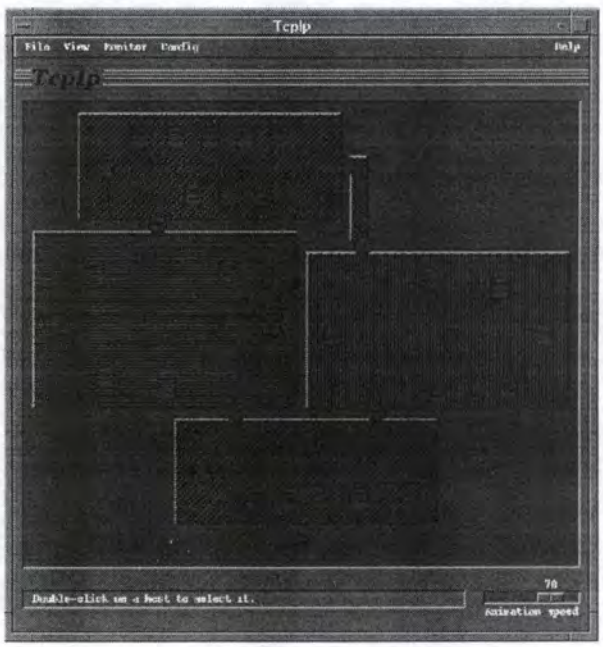


Figure 1-1 : Interface du didacticiel TcpIp dans sa version initiale.

1.2.3. Un nouveau type de sous-réseau

A partir du didacticiel TcpIp, nous avons ajouté un nouveau type de sous-réseau dans l'Internet. Ce nouveau type de sous-réseau est basé sur le protocole ATM. Or, comme la configuration de la version originelle de TcpIp est rigide, nous avons décidé de créer une interface personnalisable par l'utilisateur. Le principe de cette interface est de représenter quatre sous-réseaux et trois lignes louées. L'utilisateur peut assigner le type de sous-réseau qu'il désire voir présent dans l'Internet, et ce pour tous les sous-réseaux représentés. Les lignes louées, quant à elles, restent figées dans la configuration.

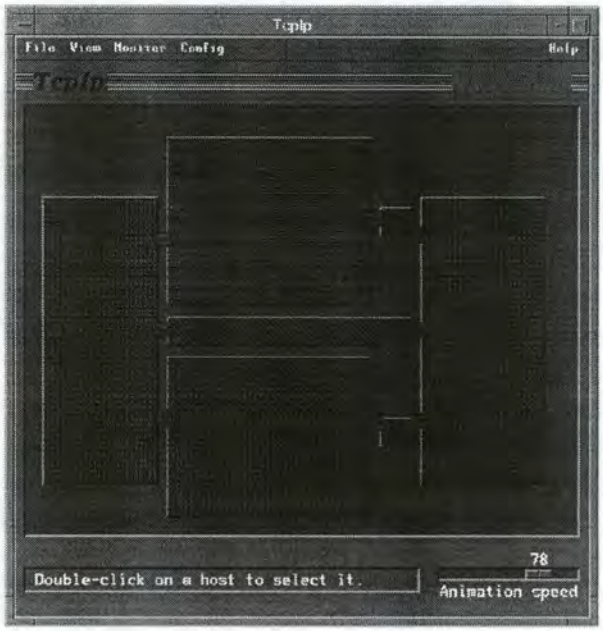


Figure 1-2 : Nouvelle interface du didacticiel TcpIp.

Cette nouvelle interface est présentée à la *Figure 1-2* avec trois sous-réseaux de type ATM, un sous-réseau à commutation par paquets et trois lignes louées.

Outre le changement d'interface, et contrairement à ce qui est valable pour les types de sous-réseaux déjà présents initialement dans TcpIp, la simulation d'une communication, au sein d'un réseau ATM, ne se limite pas au niveau logique de la couche *"réseau"*. Nous décrivons les mécanismes de communication jusqu'au niveau physique. De plus, les concepts du protocole ATM sont détaillés, pour chaque couche, de manière beaucoup plus approfondie que ce qui était fait pour les autres protocoles dans la version originelle du didacticiel.

1.3. Le projet COLOS.

Colos, qui signifie *COnceptual Learning Of Science*, est un projet fondé en 1988 par Zvonko Fazarinc dans le cadre du projet européen COMETT. A la fin de ce projet européen, les membres de Colos décidèrent de continuer leur entreprise commune et de maintenir le projet en vie. Actuellement, le groupe rassemble dix groupes de recherche répartis dans sept pays européens. La répartition géographique des pays membres s'établit comme illustré à la *Figure 1-3.* Les centres d'intérêt poursuivis par Colos sont répartis entre les divers groupes suivant leurs préférences et leurs besoins. Les disciplines auxquelles appartiennent les différents projets sont : physique, chimie, informatique, électronique, génie électrique et génie mécanique [MUL95].

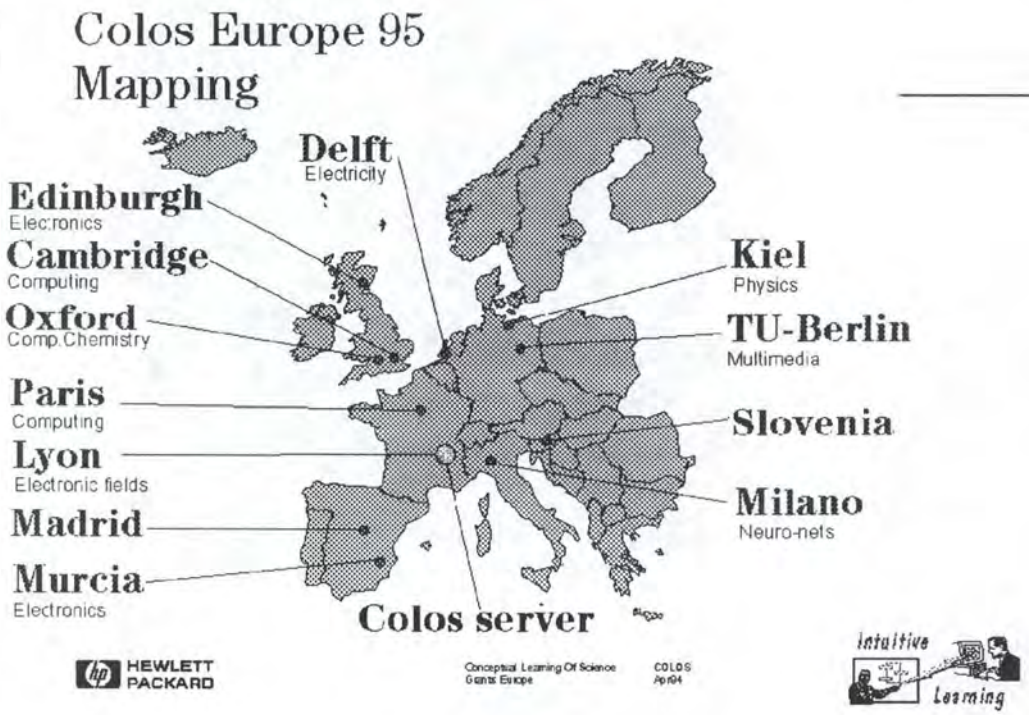


Figure 1-3 : Répartition géographique des membres du projet Colos en 1995.

Colos encourage l'apprentissage intuitif et ouvre des opportunités pour des collaborations entre universités en matière de recherche et d'enseignement. Ces universités ont formé des consortiums et des projets visant à avoir un effet profond sur la manière dont l'enseignement supérieur est mené.

La prémisse à la base de Colos est de réaliser des choses compréhensibles à partir de modèles physiques intuitifs. C'est pourquoi beaucoup de logiciels développés par Colos utilisent la simulation comme valeur ajoutée pour la présentation de processus, de modèles et de concepts importants.

1.3.1. La philosophie Colos

La technologie des ordinateurs modernes offre aux professeurs les moyens de pallier l'abstraction de modèles mathématiques complexes et de rendre les sciences plus tangibles pour les étudiants. La formidable puissance des stations de travail d'aujourd'hui et de leur environnement permet d'imiter, dans une certaine mesure, la nature dans ses principes fondamentaux. Elle offre ainsi aux étudiants des logiciels faciles à utiliser et hautement interactifs pour les guider vers une compréhension intuitive et profonde des phénomènes naturels complexes.

La configuration type sur laquelle se basent les membres de Colos pour développer leurs applications est constituée de stations HP. Les standards comme Xwindow et Motif servent de base à l'implémentation et garantissent la portabilité.

Cependant, en suivant cette approche, Colos doit être attentif aux besoins réels des différents groupes d'utilisateurs souvent équipés de PC's standards et d'environnements logiciels non compatibles. C'est pourquoi différentes activités parallèles ont été envisagées et réalisées localement pour combler ces besoins. Parmi celles-ci, on note le transfert d'applications sur *LINUX*, le re-développement sous *DOS* et le développement d'applications sur *SmallTalk* ou *Java*.

La prochaine phase du projet va voir les applications et les outils développés jusqu'à présent subir une évaluation extensive.

Ces nouveaux outils en appellent à une révision des méthodes d'enseignement. En effet, l'approche intuitive des concepts de base permet aux étudiants de comprendre la réalité de la nature et du monde sans d'abord devoir en comprendre les équations. Celles-ci peuvent être introduites plus tard, en les présentant comme ce qu'elles sont vraiment, c'est-à-dire un modèle s'efforçant de représenter la réalité, et non comme souvent imaginé par les étudiants, une loi que la nature suit.

1.3.2. Quelques scénarios Colos

I. Infrastructure de cours

Le cours est présenté dans une salle de cours équipée d'une station de travail et d'un projecteur vidéo ou d'un certain nombre de moniteurs reliés au réseau. Le professeur utilise un ordinateur pour montrer des phénomènes complexes comme des animations provenant de programmes de simulation tournant en temps réel. L'interactivité et la simulation en temps réel constituent autant de facteurs essentiels permettant au professeur de réagir immédiatement aux questions des étudiants ou de personnaliser son cours.

II. Apprentissage individuel

En apprentissage individuel, les étudiants ont accès à des stations de travail sur base d'un self-service. Le logiciel utilisé par le professeur dans la salle de cours et par les étudiants sous la direction de celui-ci, est maintenant accessible librement par les étudiants pour y suivre leur propre schéma mental, pour comprendre de nouveaux aspects et rentrer plus profondément dans le phénomène étudié. Cet environnement va permettre de développer certaines habilités reposant

sur les concepts acquis. Une fois ces simulations scientifiques incluses dans du texte *html* et rendues accessibles via un formulaire interactif sur le World Wide Web, les créations Colos peuvent jouer un rôle important dans l'apprentissage à distance.

1.3.3. Aspects conceptuels de java

Le langage de programmation *JAVA* a été conçu de manière à permettre le développement d'applications dans un contexte d'environnements hétérogènes et distribués. Parmi les défis relevés par Java, on peut citer la possibilité de transférer des applications pouvant tourner sur n'importe quelle plate-forme de manière sûre via l'utilisation des protocoles tels que HTTP (*HyperText Transport Protocol*) et FTP (*File Transfer Protocol*). Java est décrit comme un langage simple, orienté-objet, robuste, sûr, portable, performant et consommant peu de ressources réseau. Les applications Java pourront accéder à des objets à travers le réseau via les URL's (*Uniform Resource Locator*) avec la même facilité que si les objets étaient situés sur le système local. De plus, tout code Java peut être interprété de façon neutre par rapport à l'architecture sur laquelle il est exécuté ce qui rend les applications Java idéales pour tourner dans un environnement hétérogène tel qu'Internet. On peut voir que les caractéristiques offertes par Java en font un langage tout à fait approprié pour développer des applications comme celles développées dans le projet COLOS.

Chapitre 2

Le design des didacticiels

Le développement d'un logiciel demande de la part du concepteur une rigueur et une méthodologie toute particulière. C'est d'autant plus vrai lorsque le logiciel va servir à l'enseignement assisté par ordinateur. Ce chapitre va décrire une méthodologie extraite de l'ouvrage [CRO90] et le chapitre 4 expliquera la manière dont nous avons appliqué les principes de cette méthodologie à notre propre didacticiel.

2.1. La base d'un didacticiel

Une démarche assez naturelle lorsqu'on veut disposer de logiciels éducatifs consiste à se tourner vers les enseignants et les formateurs qui conjuguent a priori les talents essentiels : ils connaissent leur discipline (contenus) et savent comment l'enseigner (didactique et pédagogie). La démarche que nous allons décrire ci-après est assez théorique et la méthode que nous avons utilisée pour réaliser notre logiciel est décrite dans le paragraphe VIII de ce chapitre. Par contre, nous nous sommes basés sur bon nombre de concepts parmi ceux qui seront détaillés ci-dessous pour choisir les interfaces et concevoir notre logiciel. Les choix que nous avons posés sont expliqués dans le chapitre 4.

Par la suite, la qualité professionnelle des enseignants et des formateurs est souvent jugée comme une garantie de la qualité des logiciels qu'ils conçoivent. Cependant, cette qualité n'est pas suffisante ; il faut une méthode.

La technique de l'enseignement, dans ce qu'on appelle le "face-à-face pédagogique" n'est pas transposable telle quelle à la conception d'auxiliaires pédagogiques interactifs. Pire encore, l'expérience montre que chaque fois qu'on s'acharne à une telle transposition, on aboutit à des résultats plus que décevants. Il faut en effet inverser la perspective : substituer à une *logique de l'enseignement* dans laquelle l'enseignant et le formateur supportent la plus grande partie de l'activité, une *logique de l'apprentissage* où l'initiative échoit à l'élève qui contrôle activement

son propre apprentissage au moyen d'un auxiliaire pédagogique dont l'interactivité est la condition de possibilité. Cet auxiliaire sera spécifiquement centré sur le travail personnel et autonome. A l'élève donc d'aller chercher l'information, à lui de mettre en oeuvre des procédures de traitement et de représentation de cette information, à lui de vérifier sa progression.

2.2. La pédagogie à la base du didacticiel

Une fois la décision prise de faire un didacticiel, il faut envisager sa conception. Les méthodes sont nombreuses mais une des idées véhiculées par Kel Crossley et Les Green [CRO90] est que, le plus souvent, les concepteurs sont stérilisés et appauvris par ce qu'ils savent de l'informatique.

2.2.1. Les conditions préalables à la création d'un didacticiel

L'idée de base sur laquelle Crossley et Green s'appuient pour le développement de logiciels éducatifs est la suivante : " *penser l'élève comme sujet actif qui construit ses propres savoirs.* " [CRO90]

Pour créer un logiciel éducatif, il faut réunir quatre conditions préalables. Il faut tout d'abord avoir du goût pour la création collective et le travail en équipe, parce que la conception de didacticiel requiert un travail de création collectif.

En second lieu, une véritable expertise du domaine que l'on veut traiter doit être disponible au sein de l'équipe de conception (maîtrise des contenus).

Troisième condition, un des participants doit avoir enseigné ces contenus et donc être familier des difficultés rencontrées par ceux qui sont confrontés à leur apprentissage. La conscience et la compréhension des problèmes didactiques et pédagogiques qui sont sous-jacents à un apprentissage spécifique apportent une contribution essentielle à la qualité d'un didacticiel.

Enfin, aucun membre ne doit apporter à la table commune ses préventions ou ses croyances sur ce qu'on peut faire, ou ne pas faire avec l'informatique et les ordinateurs. Nous décrirons ce qu'il en est pour notre produit dans le chapitre 4.

2.2.2. La leçon idéale

Les enseignants font partie des personnes les mieux qualifiées pour concevoir des didacticiels. Ces enseignants qui ont été formés à la pédagogie se sont vus obligés de préparer une leçon idéale : la *leçon modèle*. C'est le moment de mettre en pratique les théories de l'apprentissage. Par la suite, les enseignants n'auront plus jamais le temps de préparer leurs cours avec autant de soins.

Un didacticiel doit être conçu comme un élément qui participe à la meilleure leçon modèle qu'on puisse imaginer. Les éléments pédagogiques d'une leçon idéale qui ont été retenus pour être appliqués aux didacticiels sont les suivants :

- La découverte :
  les élèves peuvent utiliser le logiciel pour reproduire l'expérience plusieurs fois, de telle sorte qu'ils puissent véritablement déduire des principes généraux à partir d'une quantité de données suffisante.

- L'acquisition de données :
  c'est une activité de type scientifique rarement réalisée en situation de classe, souvent pour des raisons de manque d'équipement.

- La représentation graphique :
  cet élément est utilisé pour mettre en évidence des concepts que le didacticiel présente et ce n'est pas le graphisme qui est étudié.

- L'expérience réelle :
  avant l'utilisation de certains logiciels de simulation, il est recommandé vivement aux enseignants de réaliser concrètement l'expérience si celle-ci est réalisable.

2.2.3. Les rôles d'un didacticiel

Un concepteur de didacticiel doit toujours garder à l'esprit les différentes façons dont le logiciel sera utilisé. Il peut être destiné à certaines activités :

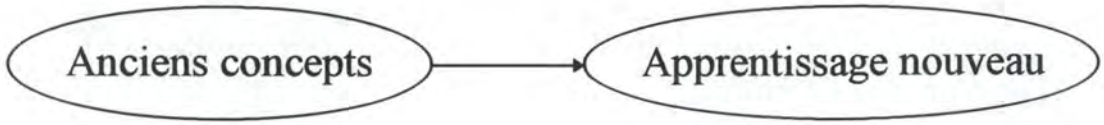
1. introduire ou réviser une notion ;
2. aider un élève à accéder à un concept particulièrement difficile ;
3. renforcer un élève en difficulté par davantage de pratique et d'exercices ;
4. être un utilitaire comme c'est le cas d'un traitement de texte ;
5. être utilisé par un groupe.

2.2.4. La loi d'Airain de l'apprentissage

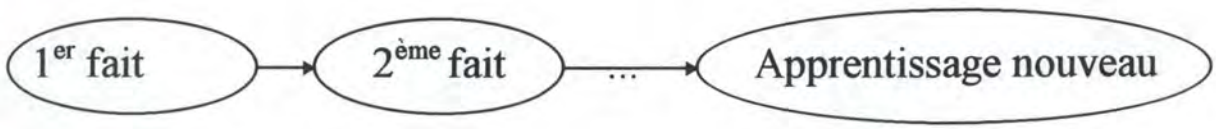
La loi d'Airain de l'apprentissage exprime le fait que les élèves acquièrent de nouveaux concepts par construction et par filtration à partir de ce qu'ils savent déjà [CRO90]. Chacun apprend en associant un nouveau concept, un nouveau fait, une nouvelle image à quelque schème cognitif déjà présent. Par exemple, lorsqu'en lisant, nous voyons un mot nouveau, nous cherchons parmi les mots environnants que nous connaissons déjà, un indice qui nous aidera à comprendre le sens de ce mot.

Dans le langage des ordinateurs, la loi d'Airain de l'apprentissage signifie qu'un didacticiel doit être *ouvert et interactif*. Chaque élève doit être capable d'explorer divers aspects d'un concept jusqu'à ce qu'il s'en souvienne et, plus encore, le comprenne.

Lorsqu'il s'agit de faire apprendre un nouveau concept, on ne peut pas se baser sur ce que chaque élève connaît déjà du sujet. Toute personne dispose en effet d'une expérience différente d'une autre. Nous ne pouvons donc pas organiser l'enseignement de la manière suivante :



Peut-être faut-il alors préparer des leçons centrées exclusivement sur le sujet abordé ? Certains enseignants et psychologues croient qu'on ne devrait enseigner que les éléments spécifiques attachés au concept précis à apprendre. Dans ce cas, le mode d'acquisition de l'élève s'ordonne ainsi :



En réalité, beaucoup de formations spécifiques peuvent s'effectuer de cette façon. Toute activité décomposable en unités élémentaires, dont chacune s'acquiert séparément, peut être organisée de telle façon que les élèves en maîtrisent successivement chaque élément.

Dans notre logiciel, nous pouvons déjà dire que c'est sur ce principe que nous nous sommes basés.

2.3. Le choix des concept-clés

L'ordinateur peut mettre en valeur des concepts qui sont habituellement difficiles à enseigner. Quels concepts allons-nous choisir pour un didacticiel ? Voici quelques questions utiles à se poser pour résoudre ce problème de choix :

- Quels sont les sujets que nous avons trouvés difficiles pour les élèves ?
- Où y a-t-il beaucoup de répétitions, de tâches fastidieuses, d'activités mécaniques qui ne demandent aucune réflexion ?

Il n'y a pas de règles générales pour choisir mais la combinaison des possibilités de l'ordinateur avec des concepts difficiles fournira un didacticiel qui enrichira vraiment l'activité de ceux qui enseignent comme de ceux qui apprennent.

2.4. L'architecture d'ensemble du design

Au cours de cette étape, les aspects suivants de la conception du didacticiel seront développés :

1. Le point de vue de l'élève.
2. Le type de didacticiel retenu.
3. Les contenus spécifiques du programme d'entraînement.
4. Les actions entreprises par l'élève, par l'ordinateur, par le professeur.

2.4.1. Le point de vue de l'élève

Imaginons tous les points de vue possibles de l'élève durant son apprentissage. Nous pouvons lui offrir un point de vue unique ou différents points de vue sur le sujet du logiciel. Nous allons considérer le point de vue de l'élève sous trois angles :

- la perception du *temps* par l'élève.
- le *rôle* que l'élève joue en utilisant le didacticiel.
- le *lieu* dans lequel l'élève se pense lui-même situé.

I. Le temps

Certains didacticiels simulent le déroulement d'événements dans le temps. D'autres conservent la trace de toutes les actions de l'élève. D'autres encore permettent à l'élève d'enregistrer des phénomènes en temps réel pour pouvoir les analyser plus tard.

Les logiciels de simulation peuvent prendre en compte le problème du temps de différentes manières. Voici quelques exemples de manières d'appréhender le temps dans un didacticiel :

- accélérer le temps : descente d'un glacier sur un millier d'années en une minute.
- ralentir le temps : circulation d'information sur un réseau informatique sur quelques kilomètres décomposée en plusieurs secondes.
- Voyager dans le passé : une journée à Versailles sous le règne de Louis XIV.
- Faire un bond dans le futur : la population mondiale durant les cinquante prochaines années.
- Arrêter le temps : suspendre le trafic sur un réseau informatique pour analyser le format des données passant sur le réseau.

II. Le rôle joué par l'élève

Lorsqu'il apprend, l'élève joue toujours un rôle. Il existe bien évidemment des rôles plus intéressants à jouer que d'autres. L'élève peut être plus ou moins actif devant le didacticiel. Cependant, plus son rôle est actif, plus son attention et son intérêt pour le contenu du logiciel risquent d'être grands.

III. La position et l'environnement spatial

L'élève peut être transporté au-delà des murs de la classe, sur la scène même où se déroule le sujet du logiciel. En choisissant une position et un environnement spatial pour l'élève, nous devons prendre en considération les angles de vue sur cet environnement, depuis cette position. Nous pouvons nous baser sur l'idée suivante : essayons de mettre l'élève là où on ne peut jamais aller dans une salle de classe traditionnelle. Nous devons nous poser la question de savoir où l'élève pense se trouver dans notre design.

2.4.2. Le type de didacticiel

Il y a encore peu de temps, tous les logiciels éducatifs étaient de type exercices et entraînements pratiques. Puis vinrent les jeux d'arcade. Les auteurs-concepteurs doivent dépasser ces deux stéréotypes et trouver d'autres modèles pour leur design. Voyons les alternatives possibles :

- le *tutoriel* : enseigne quelque chose,
- la *simulation* : modélise un processus ou une situation,
- *l'outil* : facilite la réalisation d'une tâche,
- le *jeu* : canalise la fonction ludique, permet d'apprendre sans appréhension,
- le *générateur de didacticiels* : génère de nouveaux didacticiels avec la même structure mais un contenu nouveau,
- les *exercices et entraînements pratiques* : reproduisent longuement les mêmes activités.

La plupart des applications conçues aujourd'hui combinent des éléments de ces différentes catégories de logiciels.

2.4.3. Les contenus du cursus d'enseignement

Un cursus d'enseignement amalgame des processus et des contenus. Nous devons inclure dans l'architecture d'ensemble de notre design toutes les caractéristiques d'un bon cursus.

- impliquer la participation de l'élève au processus de son propre apprentissage,
- s'assurer que les faits sur lesquels nous nous basons sont absolument exacts,
- n'utiliser les adverbes et les adjectifs qu'avec modération dans les textes du didacticiel à cause de leur caractère subjectif,
- se souvenir que le didacticiel ne doit pas constituer la totalité de l'expérience d'apprentissage de l'élève sur le domaine choisi.

2.4.4. La table des responsabilités - qui fait quoi ?

Certains concepteurs anticipent un élève passif, absorbant la connaissance et qui n'est jamais tenté d'éteindre la machine. Avec leurs logiciels, c'est l'ordinateur qui "s'amuse". Or ce qui est unique avec les ordinateurs, c'est leur capacité à être interactifs.

Un bon design est celui dans lequel l'élève se sent en mesure de contrôler son propre apprentissage. C'est l'élève et non l'ordinateur qui doit s'amuser.

Il s'agit à ce stade de l'architecture d'ensemble du design, de décider qui fait quoi, entre l'élève, l'ordinateur et l'enseignant.

2.5. Dessiner l'écran-clé

Il faut commencer par concevoir l'écran-clé. En effet, avec le logiciel, l'élève passera le plus de temps à utiliser cet écran. C'est la partie la plus importante du logiciel ! Mais c'est aussi sur cet écran que nous rencontrerons, en tant que concepteur, les problèmes pédagogiques les plus difficiles. Nous laisserons de côté pour le moment, tous les écrans jugés plus faciles.

2.5.1. Les ingrédients de l'écran-clé

Un écran-clé intègre plusieurs choses.

- L'écran-clé est la *base* à partir de laquelle l'élève explorera le concept au coeur du didacticiel. L'élève conduit son exploration à partir de cet écran ; il se transforme en réponse aux actions de l'élève ; l'élève va vers un autre ou d'autres écrans pour traiter un concept secondaire et revenir ensuite à l'écran-clé. C'est vraiment le quartier général.

- L'écran-clé est le *lieu de l'action* où l'élève agit sur l'information donnée et explore les options offertes par le logiciel.

- L'écran-clé est le *lieu de la rétroaction* où s'affichent les réactions aux instructions et aux explorations de l'élève. La rétroaction renforce l'apprentissage.

- L'écran-clé est le *lieu de l'intégration* des idées.

- L'écran-clé est *l'illustration de l'expérience* qui donne à l'élève une image centrale à s'approprier. L'écran-clé devient la clé de voûte de cet édifice d'idées, de sensations, de représentations qui se forment dans l'esprit de l'élève une fois la leçon apprise.

2.5.2. L'écran-clé et l'ordinateur

Un vieux proverbe chinois dit :

> *"J'entends et j'oublie,*
> *Je vois et je me souviens,*
> *Je fais et je comprends."*

Un ordinateur nous permet d'ordonner ces trois caractéristiques dans un écran-clé. L'accent doit être mis sur le verbe *faire* : les commandes que nous créons dans le didacticiel sont réellement des ordres que donne l'élève à l'ordinateur.

Faire dans le contexte de la conception d'un écran-clé, implique les actions suivantes de l'élève et les actions mentales qui doivent précéder chaque action :

Sélectionner	"Voici ce que j'ai choisi"
Indiquer	"Voici ce que je regarde"
Déplacer le curseur	"Voici le chemin que je suis"
Entrer les données	"Voici l'information que je donne"
Choisir une commande	"Voici ce que je veux que tu fasses"

Attendre	"J'attends"
Partir	"J'en ai assez de ce logiciel"

2.5.3. L'élève, l'écran-clé et l'ordinateur

Lorsqu'on conçoit un écran-clé, on peut, et même on devrait, prévoir suffisamment de dessins à mettre dans le logiciel. Un dessin vaut mieux qu'un long discours pour expliquer, enrichir, donner des informations générales. De façon croissante pour des générations qui grandissent avec la télévision, l'image est une source majeure d'information.

L'apprentissage peut être considérablement renforcé par l'utilisation de sons et d'effets visuels. Ils constituent le principal moyen de manifester la réaction du logiciel à l'action de l'élève, c'est-à-dire le résultat affiché de ce qui a été fait jusque là.

Il est probable que la moitié environ des élèves pensent essentiellement en termes visuels ou concrètement ; les autres, généralement les plus forts, sont à l'aise avec les idées abstraites qu'on manie bien plus facilement avec le langage. L'écran-clé doit répondre à ces deux types d'élèves.

Le verbe qui commence chacune des phrases suivantes suggère la manière dont le logiciel devrait réagir aux actions de l'élève.

- *Résumer* avec des images, des tableaux, des graphiques en réponse aux choix de l'élève et aux données acquises.
- *Mesurer des paramètres* avec des jauges, des échelles.
- *Attirer l'attention* par l'animation.
- *Créer une atmosphère* souvent renforcée par la couleur.
- *Coder* avec des couleurs.
- *Intégrer* globalement, souvent avec une image.
- *Mettre en garde*, corriger ou conseiller par l'envoi d'un message.
- *Informer* par un texte simple.
- *Offrir des options* par des commandes ou des sélections.

2.6. Construire la succession des états de l'écran-clé

Il faut maintenant développer dans les moindres détails, l'expérience interactive d'apprentissage. C'est maintenant qu'il faut penser à ce que l'élève verra ou entendra très précisément après avoir fait quelque chose dans l'écran-clé. Il faut donc concevoir la succession complète des états possibles de l'écran-clé.

A mesure que l'on compose la séquence des états de l'écran-clé, il faut penser à chacune des actions que l'élève pourra faire et à chacune des réactions de l'ordinateur aux actions de l'élève. Il faut décider et décrire les conditions sous lesquelles chaque commande peut être utilisée.

Une fois la séquence des écrans-clés terminée et les décisions pédagogiques sur les actions de l'élève et les réactions de l'ordinateur prises, il nous reste à ébaucher les autres écrans dont notre logiciel a besoin. Pour cela, nous devons suivre la même procédure que celle utilisée

pour l'écran-clé. Il s'agit donc de dessiner l'écran, déterminer la séquence d'états de cet écran et lister les commandes applicables. Pour ces dernières, il convient de spécifier les conditions dans lesquelles elles peuvent être déclenchées ainsi que les réactions de l'ordinateur à celles-ci.

2.7. Après le design

Une fois que le design est terminé, l'implémentation peut alors seulement commencer. C'est à ce niveau que l'informaticien entre en jeu. Idéalement, il ne devrait pas participer à la phase de design, ou à tout le moins influencer le moins possible celui-ci par la connaissance qu'il a de l'informatique, de la programmation et de ses limitations.

2.8. La méthode utilisée

Nous venons de décrire une méthode qui s'applique aux didacticiels. Pour le didacticiel que nous avons développé, nous nous sommes basés sur cette méthode ainsi que sur une approche de développement "*par prototypage transformationnel*" présentée au cours de *méthodologie de développement de logiciels* de Eric Dubois[DUB94]. Cette méthode illustrée à la *Figure 2-1* ci-dessous, se base d'une part sur une méthode transformationnelle et d'autre part sur la méthode par prototypage.

L'approche transformationnelle consiste à suivre une démarche assez linéaire qui part des spécifications fonctionnelles et qui aboutit à l'implémentation en passant par la conception. L'idée sous-tendant cette démarche est d'éviter les tests en s'assurant du caractère correct des documents transmis d'une étape à l'autre.

L'approche par prototypage est celle qui est assez suivie dans le monde industriel. Elle consiste à développer directement en implémentation, un prototype du logiciel à réaliser. Ce prototype va être progressivement élargi pour intégrer toutes les caractéristiques mentionnées dans le cahier des charges. Chaque élément implémenté est testé avant de passer à la suite de l'implémentation.

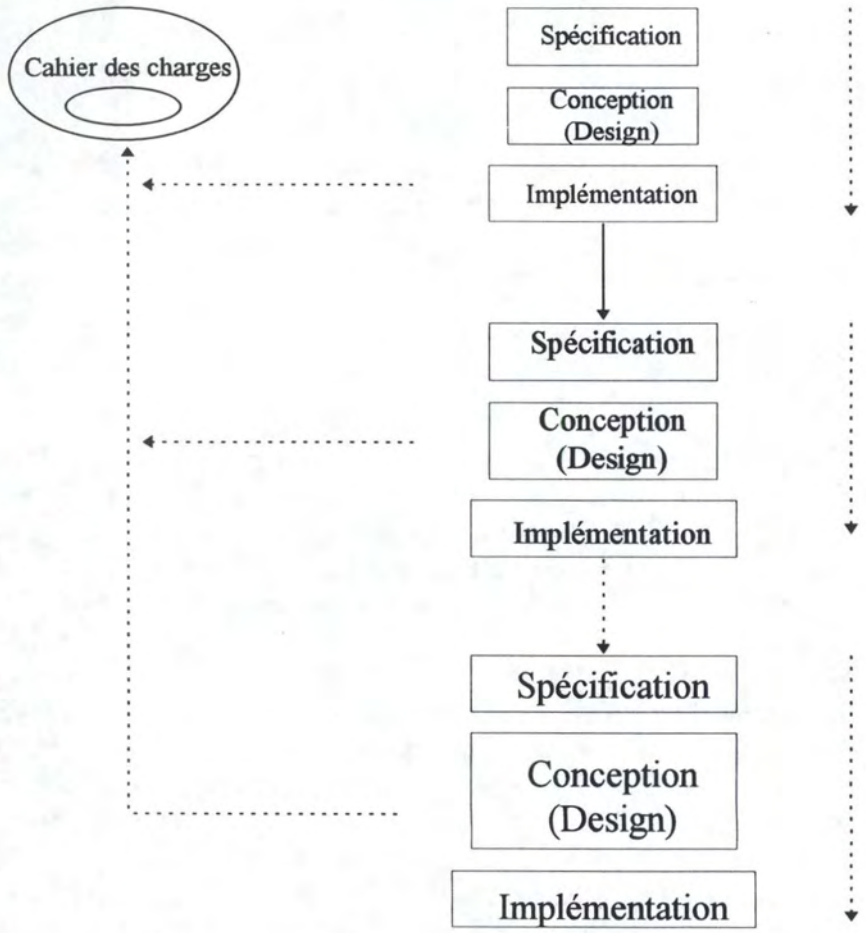


Figure 2-1 : Approche par prototypage transformationnelle

Chapitre 3

Caractéristiques de base d'ATM

Le chapitre que nous abordons ici passe en revue les caractéristiques de base du protocole ATM. Il s'agit donc de décrire la matière sur laquelle le didacticiel va être construit. La découpe proposée calque celle qui se retrouve dans le protocole lui-même. Les généralités sont abordées en premier lieu et ensuite, chaque couche du protocole est détaillée avec ses fonctions.

3.1. Généralités

3.1.1. Un protocole asynchrone

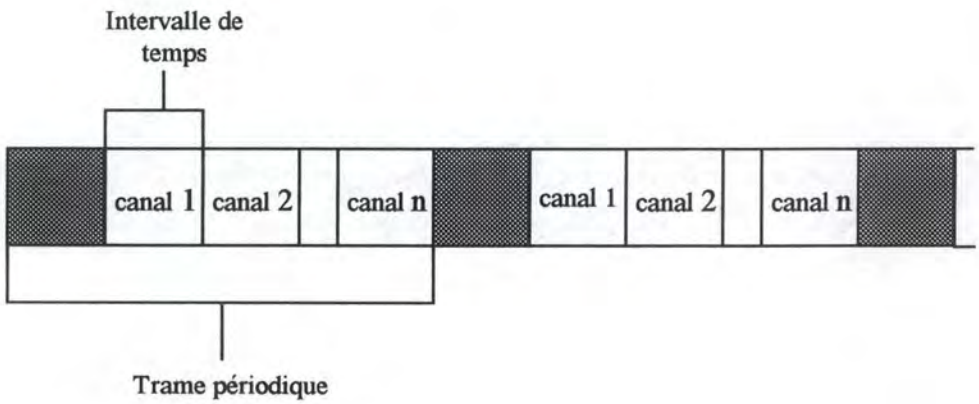
ATM signifie *Asynchronous Transfer Mode*. Il s'agit donc d'un mode de transfert, c'est-à-dire d'une manière spécifique de transmettre et d'acheminer des informations dans un réseau. Le transfert recouvre à la fois les aspects de transmission et de commutation. Par la suite, nous utiliserons indistinctement les termes protocole et mode de transfert lorsqu'on parlera d'ATM.

Alors que les lettres T et M donnent sa signification de base à l'acronyme ATM, la lettre A dépeint la caractéristique majeure du protocole. Ce A reflète l'asynchronisme avec lequel les données sont envoyées de l'expéditeur au destinataire. En effet, comme le protocole ATM transfère des informations en mode connecté, chaque demande d'ouverture de connexion donne lieu à une allocation d'un ou plusieurs canaux. Ceux-ci ne sont bien sûr que virtuels. Dès lors, toute donnée empruntant un canal virtuel ainsi ouvert, est encapsulée dans un paquet portant l'étiquette identifiant ce canal. Ces paquets sont appelés, dans ce protocole, des cellules.

L'asynchronisme signifie donc que chaque cellule associée à un canal virtuel spécifique peut apparaître à n'importe quelle position dans le flux de cellules comme le montre la *Figure 3-1 (b)*. Ce flux émane de connexions différentes, et donc de canaux multiples, multiplexés dynamiquement. Les cellules générées au fur et à mesure des besoins réels peuvent ainsi se présenter selon une périodicité irrégulière.

Par contre, dans le mode de transfert synchrone (*STM : Synchronous Transfer Mode*), la souplesse d'allocation du débit d'une connexion est réduite. Chaque connexion en mode STM utilise des débits prédéfinis et les trames périodiques possèdent une structure rigide (cfr. *Figure 3-1 (a)*).

(a) Mode de transfert **synchrone**



(b) Mode de transfert **asynchrone**

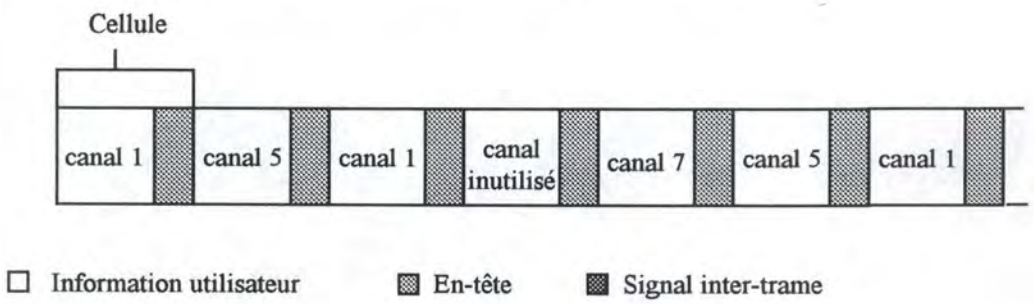


Figure 3-1 : Principes de STM et d'ATM

3.1.2. La cellule

La cellule est l'élément de base du protocole ATM. Une cellule est constituée d'un en-tête de 5 octets et d'un champ d'information de 48 octets. Sa structure est présentée dans la *Figure 3-2*.

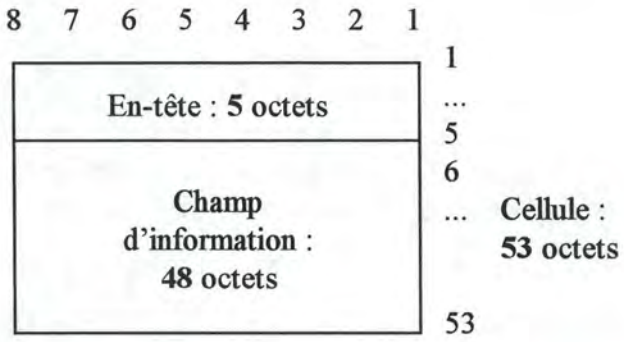


Figure 3-2 : Structure d'une cellule

3.1.3. La découpe en couche

Le modèle classique utilisé lorsqu'on parle de protocoles de réseaux est celui d'une découpe en couche. Cette modélisation en couche est utilisée pour définir l'organisation de l'ensemble des fonctions de communication. Les services assurés par ces différentes couches ainsi que les relations qui existent entre elles sont décrites dans un modèle de référence de protocole.

L'Avis I.320 de l'ITU-T fournit une description du modèle de référence utilisé pour le RNIS à bande étroite. En particulier, il introduit le concept de plans séparés permettant de différencier les fonctions à l'usager, au contrôle et à la gestion. Ce modèle constitue la base de celui du RNIS à large bande qui est décrit dans l'Avis I.321 de ITU-T.[HAN95].

I. Le modèle de référence OSI

Le modèle le plus répandu dans le monde des réseaux est le modèle OSI (*Open System Interconnection*) de l'ISO (*International Standard Organisation*) (cfr. *Figure 3-3*). Ce modèle permet de décomposer le problème des processus complexes intégrés dans la communication informatique en un ensemble de sept couches, chacune étant décrite par une spécification précise.

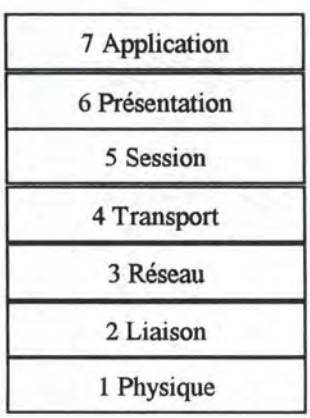


Figure 3-3 : Modèle OSI

II. Correspondance entre le modèle de référence OSI et celui du RNIS à large bande

Le modèle de référence OSI constitue une architecture logique qui définit un ensemble de principes, tels que le découpage en couche au niveau du protocole, la définition des services associés à chaque couche, les primitives de service et la notion d'indépendance entre couches. Ces fondements ont été repris pour la définition du modèle du RNIS à large bande, mais n'ont pas tous été appliqués dans leur intégralité (c'est par exemple le cas du principe d'indépendance).

Bien que la correspondance entre les couches du modèle OSI et celles du modèle du RNIS à large bande ne soit pas encore complètement établie, nous allons reprendre la tentative faite par Martin de Prycker dans [DEP92] dans l'établissement d'une correspondance.

Les relations suivantes peuvent être proposées : la couche physique en ATM est plus ou moins équivalente à la couche 1 (*Physical Layer*) du modèle OSI. La couche ATM peut être comparée principalement au bord inférieur de la couche 2 (*Data Link Layer*) du modèle OSI. La couche AAL du protocole ATM réalise l'adaptation des protocoles des couches supérieures à la taille fixe des cellules ATM. Concernant l'information du plan de contrôle, comme la signalisation, la couche AAL est équivalente à la partie inférieure de la couche 2 (*Data Link Layer*) du modèle OSI. Mais quand on regarde le plan de l'utilisateur, il est plus approprié de localiser cette couche AAL au niveau inférieur de la couche 4 (*Transport Layer*) du modèle OSI puisque l'adaptation ne se fait qu'aux extrémités du réseau.

III. Description du modèle de référence du RNIS à bande large

La *Figure 3-4* présente le modèle de référence utilisé pour le RNIS à large bande. Celui-ci comporte trois plans : plan usager, plan de contrôle, plan de gestion.

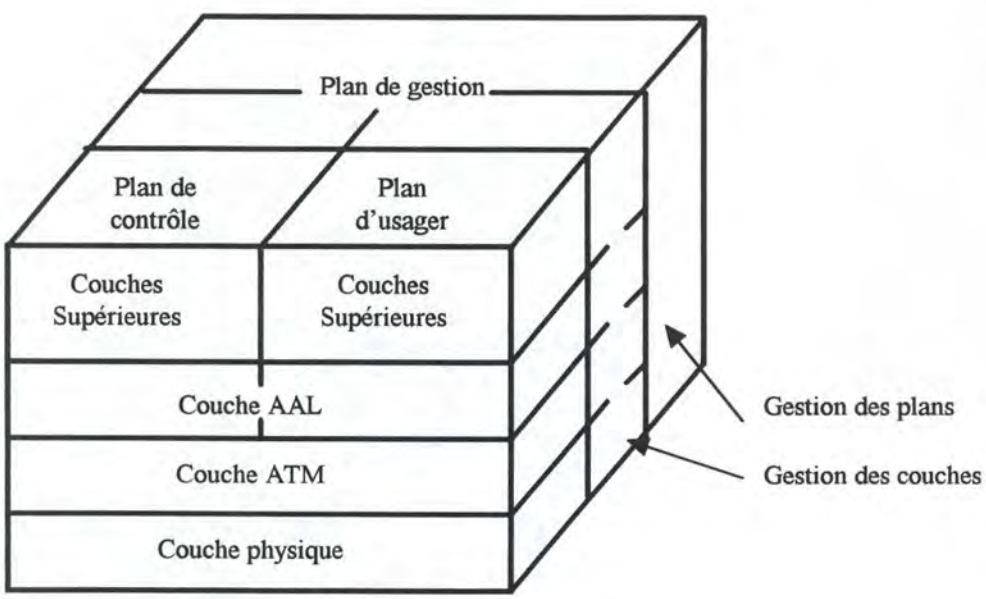
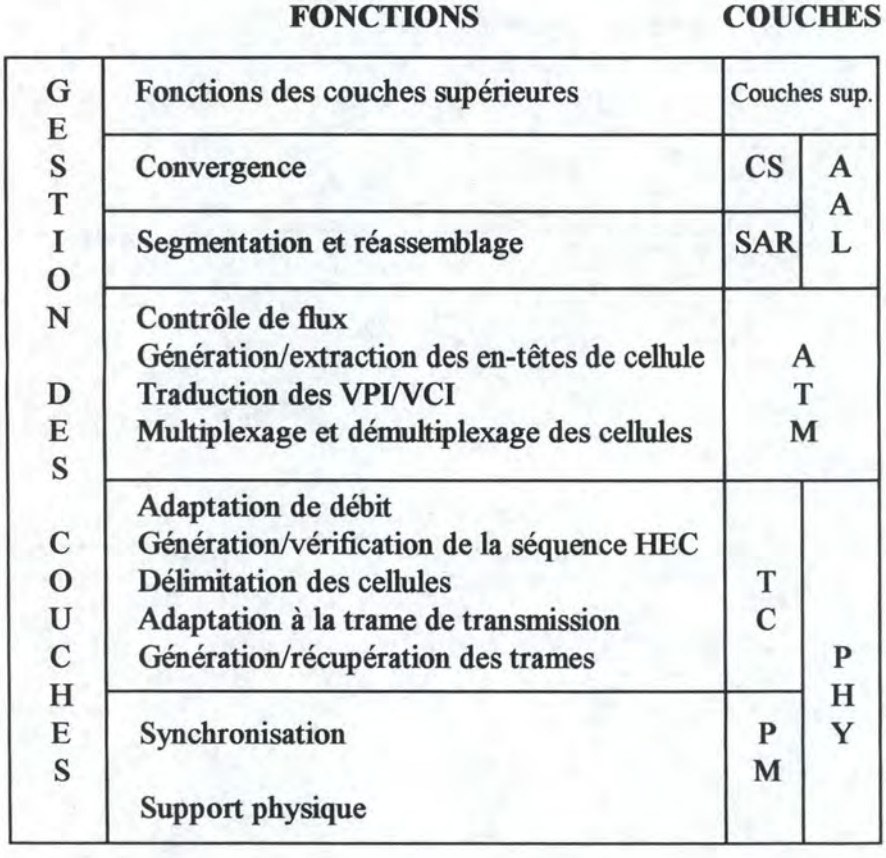


Figure 3-4 : Modèle de référence de protocole du RNIS à large bande

Le plan de gestion intègre deux catégories de fonctions correspondant à la gestion des couches et à la gestion des plans. Seul ce dernier niveau de gestion qui prend en charge l'administration du système dans son ensemble et assure la coordination entre les différents plans n'est pas organisé en couches.

La *Figure 3-5* présente les couches basses du modèle de référence du RNIS à large bande ainsi que les fonctions associées. Ces couches dites basses forment le protocole ATM. La couche AAL est subdivisée en deux sous-couches : l'une de convergence et l'autre de segmentation et réassemblage. De même, la couche physique est scindée en une couche de convergence à la transmission et une couche liée au support physique.

La signification de ces couches sera donnée ultérieurement avec l'explication des différentes fonctions associées à chacune d'elle.



CS	Convergence sublayer
HEC	Header Error Control
PM	Physical Media
SAR	Segmentation And Reassembly
TC	Transmission Convergence
VCI	Virtual Channel Identifier
VPI	Virtual Path Identifier

Figure 3-5 : Les fonctions des couches du protocole ATM

3.1.4. Le passage des PDU's

Dans ces modèles de référence qui viennent d'être présentés, une couche de niveau N communique avec son homologue distante, de niveau N également. Cette communication entre entités d'un même niveau N utilise un protocole point à point de niveau N. L'entité de niveau N exécute des fonctions à l'intérieur de la couche N et transmet à son homologue, des unités de données de protocole appelées PDU's de niveau N ou N-PDU (*N-Protocol Data Unit*). Pour communiquer entre elles, ces entités de niveau N utilisent les services fournis par les entités de niveau N-1 à qui elles passent les N-PDU's. De même une entité de niveau N offre ses services à une entité de niveau N+1.

Les (N+1)-PDU's sont considérés par les entités de niveau N comme des unités de données de service appelées SDU de niveau N ou N-SDU (*N-Service Data Unit*).

Les primitives de niveau N sont introduites pour décrire l'interface entre des couches adjacentes N et N+1. Associés à la primitive de niveau N, les N-SDU's sont acheminés de la couche N à la couche N+1 et réciproquement. Cette fonction est réalisée à l'aide du protocole de service de niveau N. La *Figure 3-6* illustre cette notion.

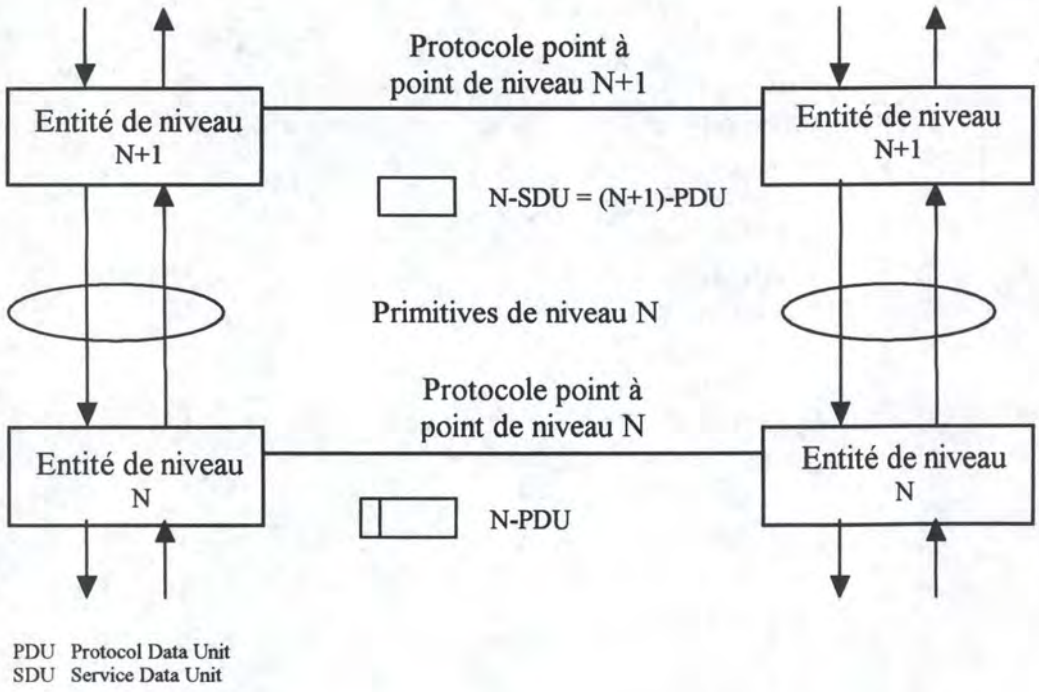


Figure 3-6 : Concept de service OSI

La *Figure 3-7* présente les relations entre les différents types d'unités de données. Un N-PDU contient des informations de contrôle de protocole de niveau N *(N-PCI : N-Protocol Control Information*) et les données utilisateur de niveau N. Ces données utilisateur sont les unités de données de service de niveau N (N-SDU) qui correspondent aux unités de données de protocole de niveau N+1 ((N+1)-PDU's). Les informations de contrôle de protocole de niveau N sont échangées et générées par les entités de niveau N.

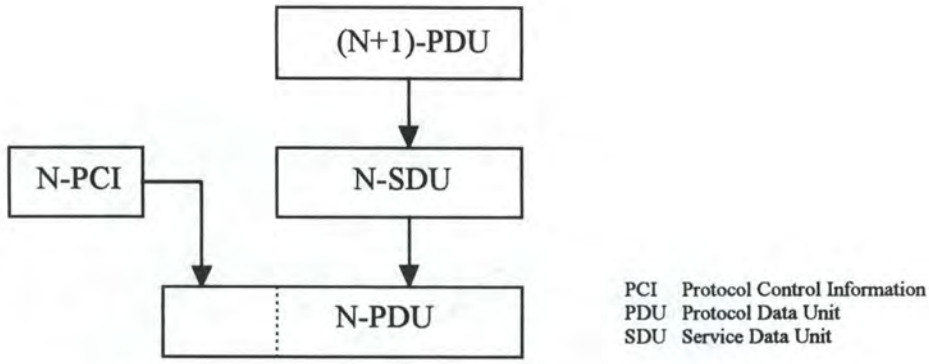


Figure 3-7 : Relations entre les types d'unités de données.

3.1.5. Architecture d'un réseau ATM

L'architecture d'un réseau ATM (voir *Figure 3-8*) peut être décrite comme suit. Le réseau ATM proprement dit est constitué d'un ensemble de noeuds, appelés commutateurs ou *switch*. Ces commutateurs supportent les couches physique et ATM du modèle de référence présenté à la *Figure 3-4*. Sur ce réseau sont connectées un certain nombre de machines terminales. Ces dernières supportent, quant à elles, les couches physique, ATM et AAL du modèle, ainsi que l'ensemble des applications qui utilisent les services du réseau ATM.

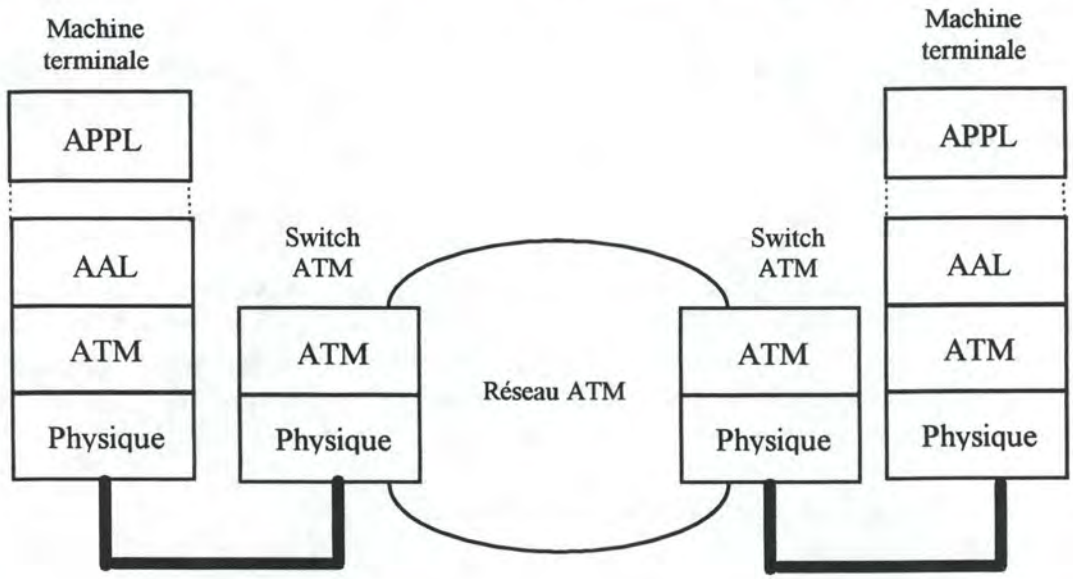


Figure 3-8 : Architecture du réseau ATM

3.2. Couche AAL

Les fonctions de la couche d'adaptation à ATM ont été présentées dans à la *Figure 3-5*. Nous allons maintenant décrire chacune d'entre elles ainsi que la couche AAL dans son ensemble.

La couche AAL est subdivisée en une sous-couche de segmentation et réassemblage (*SAR : Segmentation And Reassembly*) et une sous-couche de convergence (*CS : Convergence Sublayer*). Cette division est illustrée *Figure 3-9*.

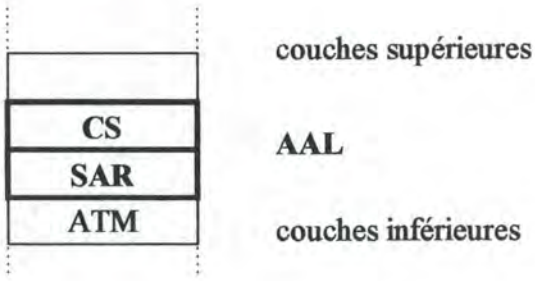


Figure 3-9 : Décomposition de la couche AAL en sous-couches

La couche AAL se situe entre la couche ATM et les couches supérieures extérieures au protocole ATM. Son rôle est d'ajuster les services fournis par la couche ATM aux exigences de la couche immédiatement supérieure. Les PDU's produits par cette couche d'adaptation sont délivrés à la couche ATM. Les entités CS et SAR de la couche AAL dialoguent avec leurs homologues distants pour fournir les services dont elles ont la charge.

3.2.1. Sous-couche de segmentation et réassemblage

Les fonctions de la couche AAL sont réparties entre les deux sous-couches la composant. La sous-couche de segmentation et réassemblage (SAR) assure, du côté émetteur, le découpage des PDU's provenant de la couche supérieure en éléments compatibles avec la taille du champ d'information des cellules ATM (48 octets). Du côté récepteur, elle reconstitue des PDU's à destination de la couche supérieure à partir du champ d'information des cellules reçues.

3.2.2. Sous-couche de convergence

La sous-couche de convergence est liée au type de service désiré par l'utilisateur. Du côté émetteur, elle reçoit les PDU's de la couche directement supérieure, ajoute des informations de contrôle propres au service utilisateur et délivre le CS-PDU formé à la sous-couche SAR. Du côté récepteur, elle extrait l'information présente dans le CS-PDU et la délivre à la couche supérieure. Elle assure les fonctions d'adaptation à ATM au niveau du point d'accès de service (*SAP : Service Access Point*) correspondant. Plusieurs points d'accès peuvent être envisagés pour les couches supérieures à partir de différentes combinaisons des sous-couches SAR et CS.

Afin de minimiser le nombre de protocoles d'adaptation à ATM, l'ITU-T a proposé une classification des services propres à la couche AAL. Celle-ci a été réalisée à partir des paramètres suivants :

- Synchronisation entre émetteur et récepteur,
- débit binaire,
- mode de connexion.

La *Figure 3-10* présente les classes de service AAL. Toutes les combinaisons possibles des trois paramètres cités ne sont pas réalistes et seulement quatre classes ont été définies sur les huit classes potentielles.

	Classe A	**Classe B**	**Classe C**	**Classe D**
Synchronisation entre émetteur et récepteur	oui	oui	non	non
Débit	constant	variable	variable	variable
Mode de connexion	connecté	connecté	connecté	non connecté

Figure 3-10 : Classification des services de la couche AAL.

3.2.3. Négociation du service

Plusieurs types de protocole AAL sont définis. Chacun d'entre eux est constitué d'une sous-couche SAR spécifique et d'une sous-couche CS spécifique. Cette classification correspond aux classes de service AAL décrites dans la section ci-dessus. Par exemple, les services à débit constant de classe A vont utiliser le protocole AAL de type 1. Cependant, aucune relation stricte entre les classes de service et les types de protocole n'est exigée. D'autres combinaisons de protocoles SAR et CS peuvent également être utilisées pour des services spécifiques.

I. AAL de type 1

Normalement, les services à débit constant (Classe A) utilisent la couche AAL de type 1 parce qu'elle reçoit et délivre des SDU's à un débit constant vis-à-vis de la couche supérieure.

Les PDU's de la couche SAR sont constitués de 48 octets comme illustré à la *Figure 3-11*. Le premier octet comporte les informations de contrôle (*PCI : Protocol Control Information*) ; tous les autres octets sont disponibles pour le CS-PDU. Les informations de contrôle se subdivisent en un numéro de séquence sur 4 bits (*SN : Sequence Number*) et un champ de protection de ce numéro de séquence (*SNP : Sequence Number Protection*) de 4 bits également. Le numéro de séquence est lui-même composé d'un indicateur de sous-couche de convergence (*C : Convergence*) sur 1 bit et d'un compteur de séquence (*SC : Sequence Counter*) de 3 bits. Le champ SNP contient un code de contrôle CRC (*Cyclic Redundancy Code*) de 3 bits qui protège le numéro de séquence et un bit de parité paire calculé sur les 7 bits précédents.

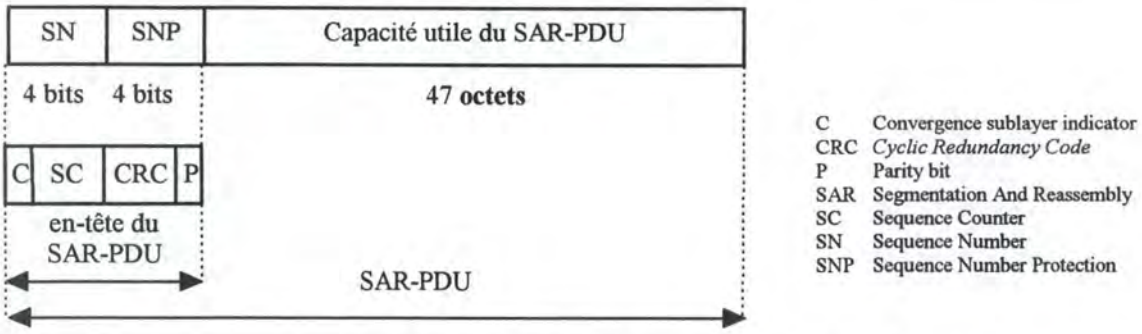


Figure 3-11 : Format du SAR-PDU utilisé par le protocole AAL de type 1

La sous-couche de convergence remplit des fonctions qui sont fortement liées au service à supporter. Parmi celles-ci, on peut mentionner le traitement de la variation du délai de propagation, le traitement des cellules perdues ou mal insérées, la restauration d'une horloge source, la transmission d'informations de structure entre la source et la destination. Le format du CS-PDU pour le protocole AAL de type 1 est illustré à la *Figure 3-12*. On peut remarquer que ce CS-PDU de la couche AAL de type 1 correspond exactement au SAR-SDU.

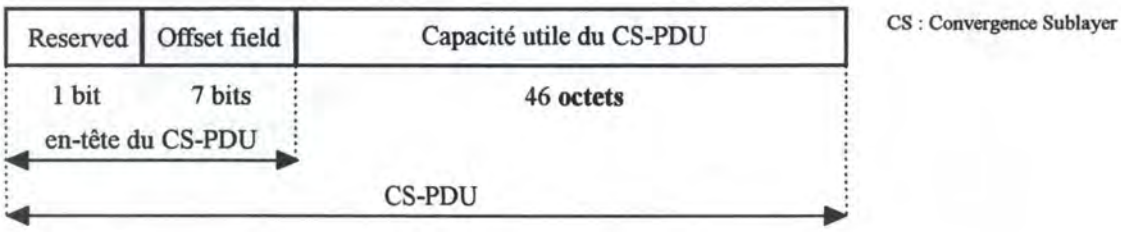


Figure 3-12 : Format du CS-PDU utilisé par le protocole AAL de type 1

II. AAL de type 2

La couche d'adaptation de type 2 est proposée pour les services à débit variable mettant en oeuvre une synchronisation entre la source et la destination (service de classe B tels qu'audio ou vidéo à débit variable). Ce type n'est pas encore bien défini. Il se pourrait également que l'AAL de type 1 soit enrichi afin de supporter les fonctions de type 2.

III. AAL de type 3/4

Le nom de cette couche est le reflet de son développement : lorsque les classes de service ont été définies, des couches AAL distinctes, appelées AAL de type 3 et AAL de type 4, ont été affectées aux services de classes C et D. Entre temps, les services ont fusionné, supportant donc les deux classes de service. Ce protocole est utilisé par exemple pour implémenter IP sur ATM.

La *Figure 3-13* présente la structure générale de cette couche d'adaptation de type 3/4. La sous-couche de convergence est éclatée en une partie commune (*CPCS : Common Part Convergence Sublayer*) et une partie spécifique au service (*SSCS : Service-Specific Convergence Sublayer*). Cette dernière est dépendante de l'application utilisatrice du service ATM et peut être vide.

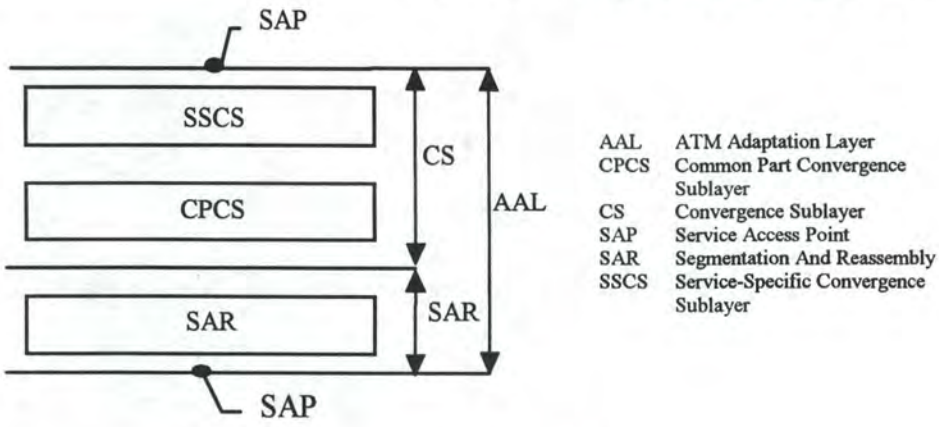


Figure 3-13 : AAL de type 3/4

Deux modes de service sont définis pour l'AAL de type 3/4, le service en mode message et le service en mode flux. Le service en mode message transporte un AAL-SDU dans un ou plusieurs CS-PDU's dont chacun peut à son tour générer un ou plusieurs SAR-PDU's.

En mode flux, un ou plusieurs AAL-SDU's de longueur fixe, sont transportés dans un CS-PDU.

La *Figure 3-14* illustre le fonctionnement du service en mode message. Le message constitue alors l'AAL-SDU. Cet AAL-SDU, avec les informations de contrôle ajoutées par la couche CS en en-tête et en terminaison, constitue le CS-PDU transmis à la sous-couche SAR. Ce CS-PDU est finalement segmenté pour être projeté dans le champ d'information des cellules.

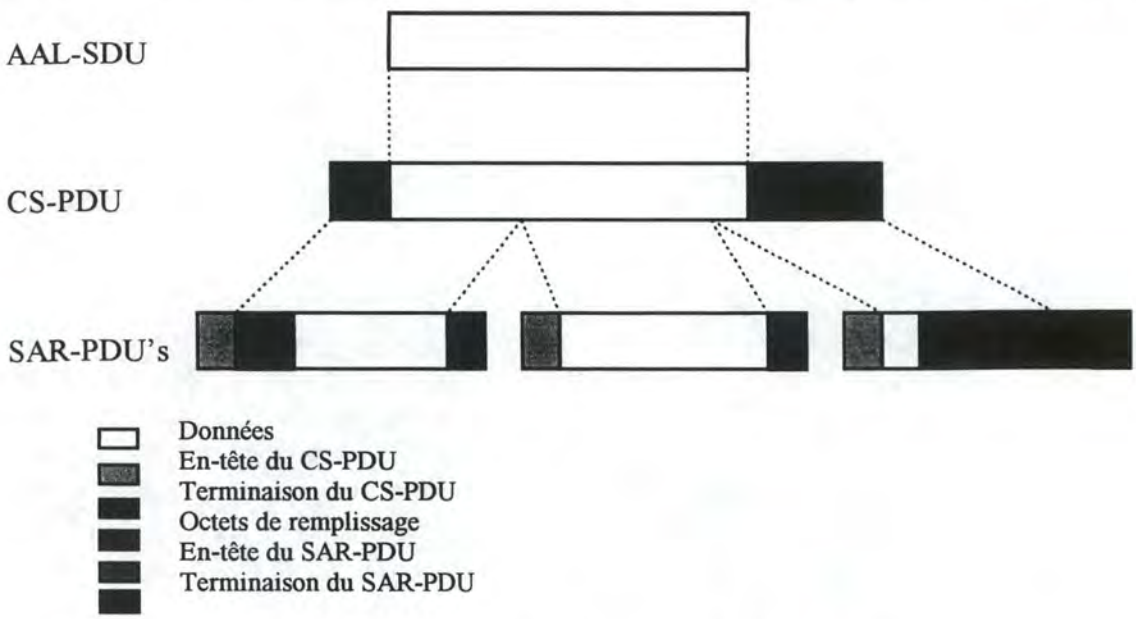


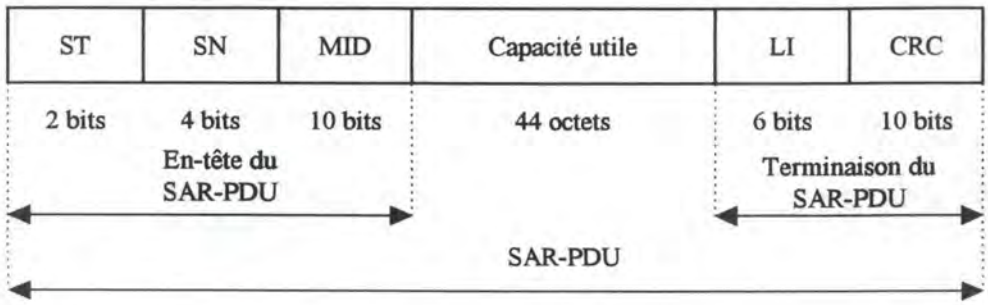
Figure 3-14 : Service en mode message

En général, les CS-PDU's sont de longueur variable. Donc quand la sous-couche SAR reçoit un PDU de ce type, elle génère des SAR-PDU's contenant jusqu'à 44 octets de données issues du CS-PDU. Cette opération nécessite un indicateur de type de segment (*ST : Segment Type*), un numéro de séquence (*SN : Sequence Number*) et un indicateur de remplissage de la capacité utile (*LI : Length Indicator*)de la sous-couche SAR.

La détection d'erreur constitue la seconde fonction de la sous-couche SAR. Cette fonction est réalisée à l'aide du champ CRC du SAR-PDU.

La troisième fonction de la sous-couche SAR est le multiplexage/démultiplexage des CS-PDU's issus de connexions AAL multiples, établies sur une même connexion ATM. A cet effet, un identificateur de multiplexage (*MID : Multiplexing Identifier*) de 10 bits est utilisé.

Pour supporter toutes ces fonctions, 4 octets sont utilisés. Donc, seuls 44 octets sur les 48 du SAR-PDU sont disponibles pour constituer la capacité utile de ce PDU. La *Figure 3-15* présente le format du SAR-PDU



CRC Cyclic Redundancy Check
LI Length Indicator
PDU Protocol Data Unit
MID Multiplexing IDentifier
SAR Segmentation And Reassembly
SN Sequence number
ST Segment Type

Figure 3-15 : Format du SAR-PDU utilisé par la couche AAL de type 3/4

La sous-couche de convergence est subdivisée en une partie commune et une partie spécifique au service. Les fonctions, la structure et la codification de la partie spécifique (SSCS) doivent encore être précisées. La partie commune (CPCS) transmet des trames de données utilisateur d'une longueur comprise entre 1 et 65.535 octets. Les fonctions associées nécessitent un CPCS-PDU comportant un en-tête et une terminaison de 4 octets chacun. De plus un champ de bourrage (*PAD : Padding*) assure l'alignement sur une frontière de 32 bits. La *Figure 3-16* présente le format du CPCS-PDU.

Le champ indicateur de partie commune (*CPI : Common Part Identifier*) est actuellement toujours à zéro. Les indicateurs de début (*Btag : Beginning Tag*) et de fin (*Etag : Ending Tag*) permettent l'association de l'en-tête et de la terminaison. Les deux champs reçoivent la même valeur numérique. Le champ BASize (*Buffer Size*) indique à l'entité homologue réceptrice la taille maximum de mémoire tampon nécessaire pour recevoir le CPCS-PDU. Les champs PAD et AL (*ALignment*) sont utilisés pour cadrer la terminaison du CPCS-PDU sur une frontière de 32 bits. Le champ longueur (Lg) est utilisé pour stocker la longueur du champ de capacité utile du CPCS-PDU.

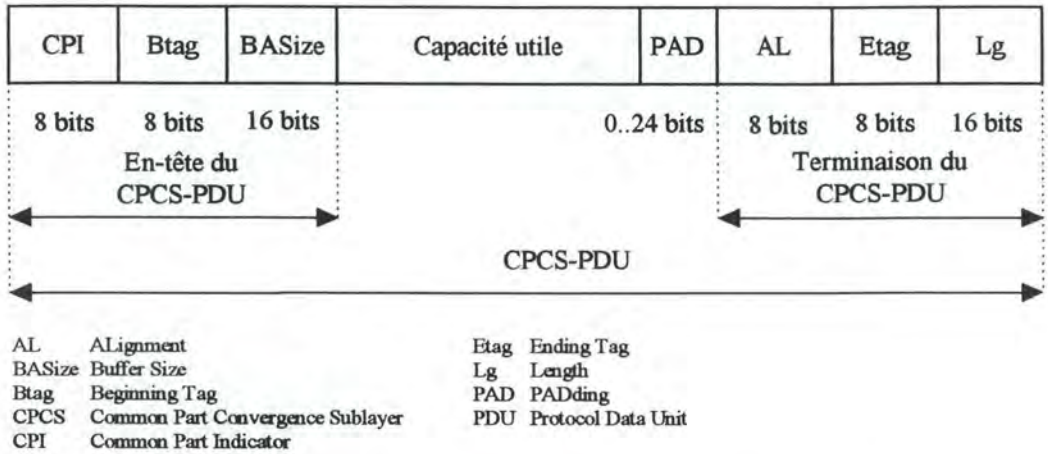


Figure 3-16 : Format du CPCS-PDU utilisé par la couche AAL de type 3/4

IV. AAL de type 5

La couche d'adaptation AAL de type 5 sera utilisée pour des applications à débit variable sans lien de synchronisation entre la source et la destination. Elle fournit des services similaires à ceux de l'AAL de type 3/4 et concerne principalement des applications orientées données. La raison justifiant la création d'un type d'AAL supplémentaire était son surdébit réduit ("un AAL simple et efficace"). Son fonctionnement est identique à celui défini pour l'AAL de type 3/4. Cependant, une différence essentielle est que l'AAL de type 5 ne supporte pas la fonction de multiplexage et ne comporte donc pas de champ MID dans son CS-PDU.

La partie commune CPCS de l'AAL de type 5 assure le transfert de trames de données utilisateur d'une longueur comprise entre 1 et 65.535 octets. De plus, un octet (*UU : CPCS user-to-user indication*) utilisé pour échanger des informations d'utilisateur à utilisateur est transmis de manière transparente avec chaque CPCS-PDU. Un champ CRC-32 est utilisé pour détecter les erreurs de bit. Un champ CPI (*Common Part Indicator*) sera utilisé ultérieurement pour des fonctions similaires à celles qu'il assure dans la couche AAL de type 3/4. Pour l'instant, il est simplement utilisé pour aligner la terminaison du CPCS-PDU sur 64 bits. Le champ Lg contient la taille de la capacité utile du CPCS-PDU. Il est également utilisé par le récepteur pour détecter les informations éventuellement perdues ou rajoutées.

Les fonctions de la CPCS nécessitent une terminaison de 8 octets dans la CPCS-PDU. De plus, un champ de bourrage (*PAD : PADding*) assure le cadrage du CPCS-PDU sur 48 octets.

Le format du CPCS-PDU utilisé par la couche AAL de type 5 est présenté dans la *Figure 3-17*.

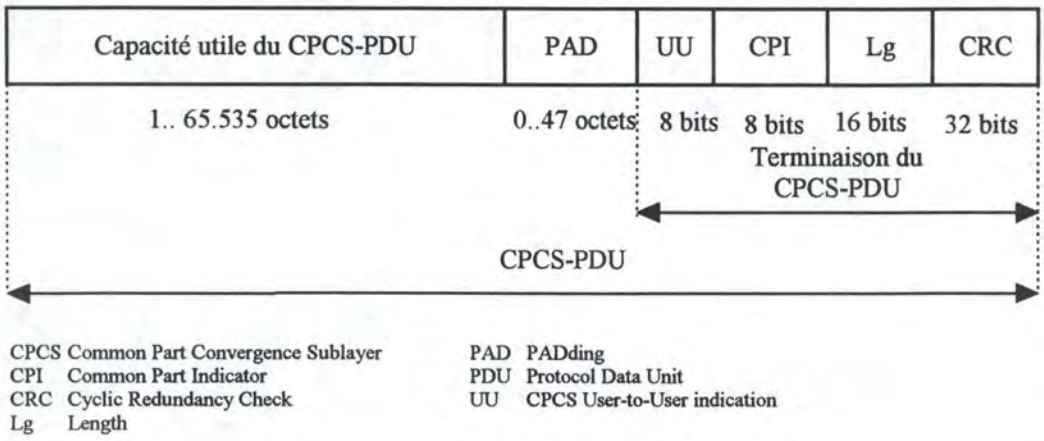


Figure 3-17 : Format du CPCS-PDU utilisé par la couche AAL de type 5

Après avoir détaillé la couche AAL dans son entièreté, nous passons à la description de la couche ATM et de ses fonctions.

3.3. Couche ATM

La notion de cellule que l'on associe au protocole ATM est en fait l'unité de base de la couche ATM. La couche ATM se trouve intercalée entre la couche AAL au-dessus et la couche physique en dessous. Ses caractéristiques sont indépendantes du support physique. Quatre fonctions ont été identifiées pour cette couche. Il s'agit du contrôle de flux, de la génération/extraction de l'en-tête de cellule, de la traduction des identificateurs de *Virtual Channel* et *Virtual Path* (*VPI* et *VCI*), du multiplexage et du démultiplexage des cellules.

3.3.1. Contrôle de flux

La fonction de contrôle de flux (*GFC : Generic Flow Control*) est uniquement définie au niveau de l'interface usager-réseau (*UNI : User-Network Interface*) du RNIS à large bande. Elle assure le contrôle du trafic ATM dans un réseau d'usager. Elle peut être utilisée pour alléger des situations de surcharge temporaire (dans le sens usager vers réseau) au niveau de l'interface.

3.3.2. Traduction des VCI/VPI

La couche ATM comprend deux niveaux hiérarchiques :

- la gestion des canaux virtuels,
- la gestion des conduits virtuels.

Commençons par définir quelques concepts utilisés dans la couche ATM :

Conduit de transmission : lien entre deux éléments du réseau fournissant la capacité utile d'un système de transmission.

Canal virtuel (*VC : Virtual Channel*) : " Concept utilisé pour décrire la transmission unidirectionnelle de cellules ATM partageant un identificateur commun unique." Celui-ci est appelé identificateur de canal virtuel (*VCI : Virtual Channel Identifier*) et est inclus dans l'en-tête de la cellule.

Conduit virtuel (*VP : Virtual Path*) : " Concept utilisé pour décrire la transmission unidirectionnelle de cellules associées à des canaux virtuels partageant un identificateur commun." Celui-ci est appelé identificateur de conduit virtuel (*VPI : Virtual Path Identifier*) et est également inclus dans l'en-tête de la cellule.

Liaison VC (resp. VP) : "Un moyen de transport unidirectionnel de cellules ATM entre un point où est attribué un VCI (resp. VPI) et un autre où cette valeur est traduite ou supprimée".

Connexion de VC (*VCC : Virtual Channel Connection*) est un regroupement de liaisons VC.

Connexion de VP (*VPC : Virtual Path Connection*) est une concaténation de liaisons VP.

La *Figure 3-18* présente les relations entre conduit virtuel, canal virtuel et conduit de transmission. Un conduit de transmission peut inclure plusieurs conduits virtuels et chaque conduit virtuel peut supporter plusieurs canaux virtuels.

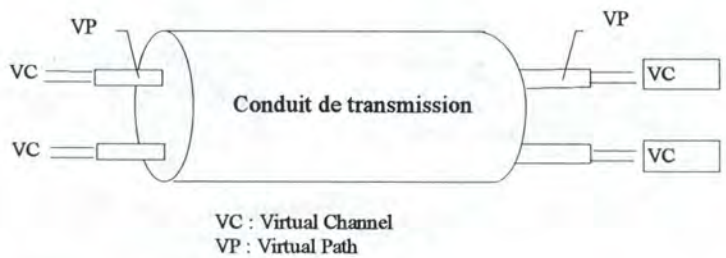


Figure 3-18 : Relations entre canal virtuel, conduit virtuel et conduit de transmission

Les VCI et VPI n'ont en général de signification que sur une seule liaison. Dans une connexion de VC ou de VP, les valeurs de ces identificateurs sont traduites à chaque passage dans un équipement de commutation de VC ou de VP. Cette traduction est réalisée, par exemple, sur base de tables de correspondance des connexions logiques entrantes et sortantes. Cette table est mise à jour lors de la phase d'établissement de chaque connexion qui transite par le noeud concerné.

Les commutateurs de VP ou brasseurs (voir *Figure 3-19*), constituent la terminaison des liaisons de VP et doivent donc effectuer la traduction entre les VPI entrants et les valeurs correspondantes en sortie, en fonction de la destination de la connexion de VP.

Les commutateurs de VC (voir *Figure 3-19*) terminent à la fois les liaisons de VC et, nécessairement, les liaisons de VP. Ils effectuent donc la traduction des VCI et des VPI.

En fonction des définitions que nous venons d'énoncer et de la *Figure 3-17* illustrée ci-dessous, nous pouvons donner l'exemple d'une connexion de VC qui est formée des couples **VPI1/VCI1** et **VPI2/VCI4**.

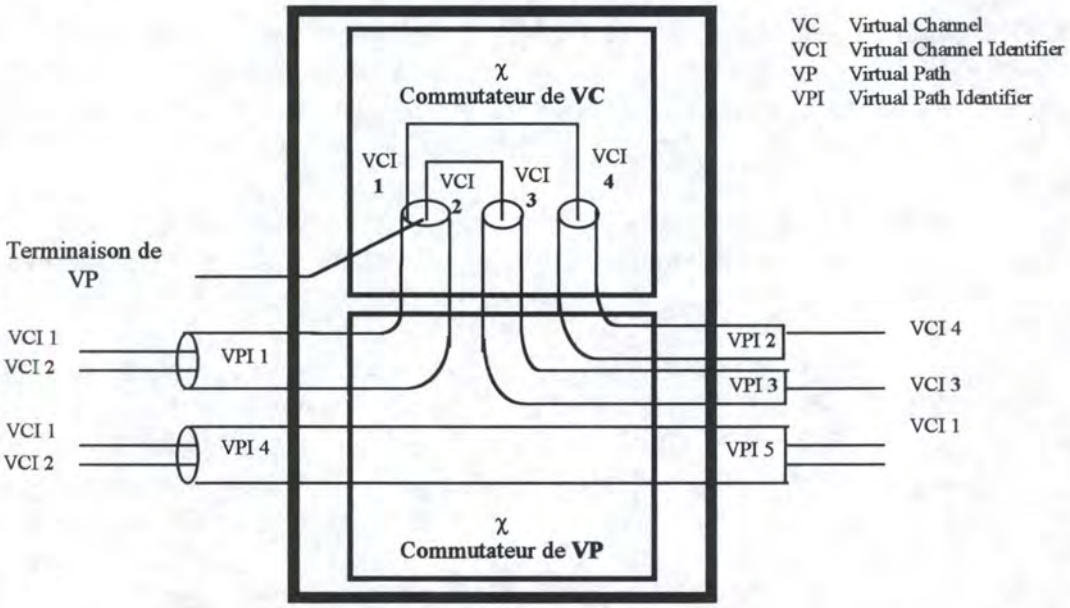


Figure 3-19 : Commutation de VC et de VP

Toutes les cellules associées à une connexion individuelle suivent le même itinéraire sur le réseau. La séquence des cellules est conservée sur toutes les connexions de VC et de VP. Par ce principe, ATM ressemble au mode traditionnel de transfert par circuits.

Une des quatre méthodes suivantes peut être utilisée pour établir une connexion de VC à l'UNI :

- Les connexions permanentes ou semi-permanentes sont établies au moment de l'abonnement. Aucune procédure de signalisation n'est donc nécessaire.
- Un VC de signalisation est ouvert en utilisant une procédure de méta-signalisation.
- L'établissement et la libération d'une connexion de VC commutée de bout en bout peuvent être réalisés par une procédure de signalisation d'usager vers réseau.
- Si une connexion de VP existe déjà entre deux UNI, une connexion de VC peut être établie/libérée dans ce VP en utilisant une procédure de signalisation d'utilisateur à utilisateur.

3.3.3. Multiplexage et démultiplexage des cellules

En émission, les cellules individuelles issues de VC et de VP sont multiplexées par l'intermédiaire de la fonction de multiplexage des cellules afin de produire un flux unique. Le résultat composite obtenu constitue normalement un flux discontinu de cellules. Côté récepteur, la fonction de démultiplexage des cellules décompose le flot de cellules entrant en autant de flux individuels correspondant à chaque canal ou conduit virtuel.

3.3.4. Génération et extraction des en-têtes de cellules

La fonction de génération et d'extraction de l'en-tête de cellule est exécutée aux points de terminaison de la couche ATM. En émission, le processus de génération ajoute au champ d'information reçu de la couche AAL, l'en-tête de cellule ATM, à l'exception du champ HEC qui lui est généré au niveau de la couche physique. Dans le sens opposé, la fonction d'extraction enlève l'en-tête de la cellule et délivre l'information à la couche AAL.

3.3.5. Le format des cellules

La cellule est l'élément de base du protocole ATM et plus particulièrement de la couche ATM. Le terme cellule est aussi utilisé au niveau de la couche physique. Une cellule est constituée d'un en-tête de 5 octets et d'un champ d'informations de 48 octets. Sa structure est présentée dans la *Figure 3-20*.

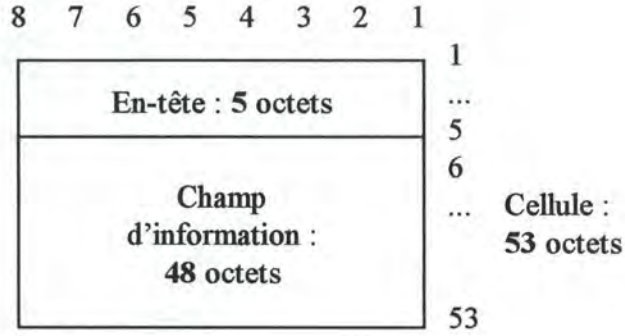
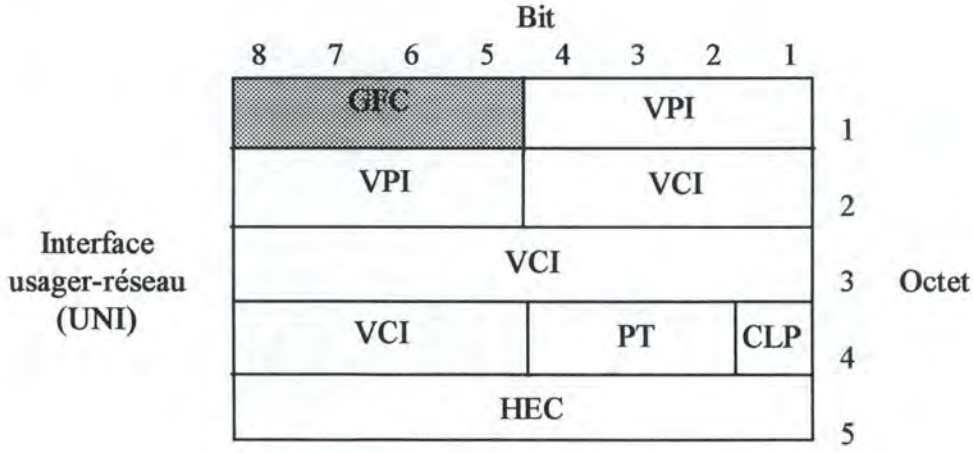


Figure 3-20 : Structure d'une cellule

I. L'en-tête de cellule

L'en-tête de cellule à l'interface usager-réseau (*UNI*) du RNIS à large bande diffère de celui de l'interface entre noeuds du réseau (*NNI*) dans l'utilisation des bits 5 à 8 de l'octet 1. Cette différence est matérialisée en grisé dans la *Figure 3-21* qui illustre les en-têtes de cellules utilisés respectivement aux interfaces UNI et NNI. A l'interface NNI, ces bits font partie du VPI, alors qu'à l'interface UNI, ils constituent un champ indépendant dédié au contrôle de flux (*GFC*).



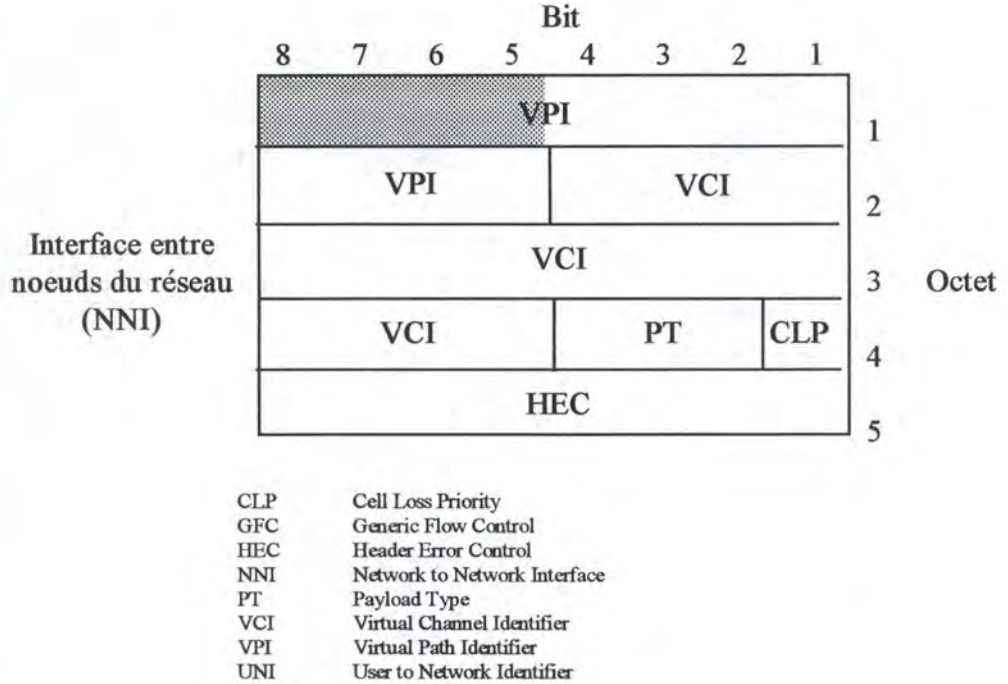


Figure 3-21 : En-tête de cellule aux interfaces UNI et NNI

Les différents champs définis dans l'en-tête de cellule n'ont aucune signification dans le cas de cellules de la couche physique. Pour la couche ATM par contre, les champs ont la signification suivante :

- GFC : (*Generic Flow Control*) ce champ signifie "contrôle du flux générique". Il intervient dans la fonction de contrôle de flux (cfr. supra). La valeur par défaut de ce champ est 0000 tant que la fonction de contrôle de flux n'est pas utilisée.

- VPI : (*Virtual Path Identifier*) ce champ signifie "identificateur de conduit virtuel". Il est utilisé pour le routage. La longueur de ce champ diffère à l'interface usager-réseau (8 bits) de celle entre noeuds du réseau (12 bits). Les quatre bits supplémentaires présents au NNI permettent un routage étendu au sein du réseau. Des valeurs prédéfinies du champ VPI sont utilisées à des fins spécifiques. Par exemple, tous les bits de ce champ sont mis à 0 dans le cas de cellules non assignées à un VP.

- VCI : (*Virtual Channel Identifier*) ce champ signifie "identificateur de canal virtuel". Utilisé en liaison avec le champ VPI, il permet le routage d'une cellule. Un champ de 16 bits est utilisé pour cet identificateur. Il existe également certaines valeurs prédéfinies pour ce champ. Par exemple, la valeur 1 est réservée pour le canal de méta-signalisation dans chaque conduit virtuel.

- PT : (*Payload Type*) ce champ signifie "type de capacité utile". Trois bits de l'en-tête sont utilisés à cet effet d'identification du type de la capacité utile. Parmi ceux-ci, il y en a un qui est réservé à l'indicateur de congestion. Ce bit peut être modifié par n'importe quel élément du réseau qui est en situation de congestion pour en avertir l'utilisateur final.

- CLP : (*Cell Loss Priority*) ce champ signifie "préférence à l'écartement". Il s'agit d'un seul bit utilisé pour indiquer explicitement la préférence à l'écartement des cellules. Si ce bit est à 1, la cellule est susceptible d'être éliminée en fonction des conditions de fonctionnement du réseau. Dans l'autre cas (CLP = 0), la cellule possède une priorité haute et les ressources nécessaires au niveau du réseau doivent donc lui être affectées.

- HEC : (*Header Error Control*) ce champ signifie "contrôle d'erreur sur l'en-tête". Ce champ fait partie de l'en-tête de cellule mais il n'est pas utilisé par la couche ATM. Il contient le code de contrôle d'erreur qui est traité par la couche physique

II. Le champ d'information

Le champ d'information des cellules générées par la couche ATM n'est pas composé uniquement d'informations utilisateur. Ces 48 octets sont encore décomposés en champs de contrôle et de données utilisateur. Cette décomposition dépend du service demandé par l'utilisateur et est prise en charge par la couche AAL. La couche ATM reçoit les AAL-PDU et les place dans le champ information des cellules en émission. Côté réception, ce champ est extrait et transmis à la couche AAL.

Il nous reste à examiner la couche physique en détail pour terminer le passage en revue des couches du protocole ATM.

3.4. Couche PHYSIQUE

Selon le modèle de référence décrit à la *Figure 3-4*, la couche physique est subdivisée en une sous-couche de média physique (*PM : Physical Medium*) et une sous-couche de convergence au niveau transmission (*TC : Transmission Convergence*). Celles-ci sont présentées dans la *Figure 3-22* ainsi que les fonctions associées.

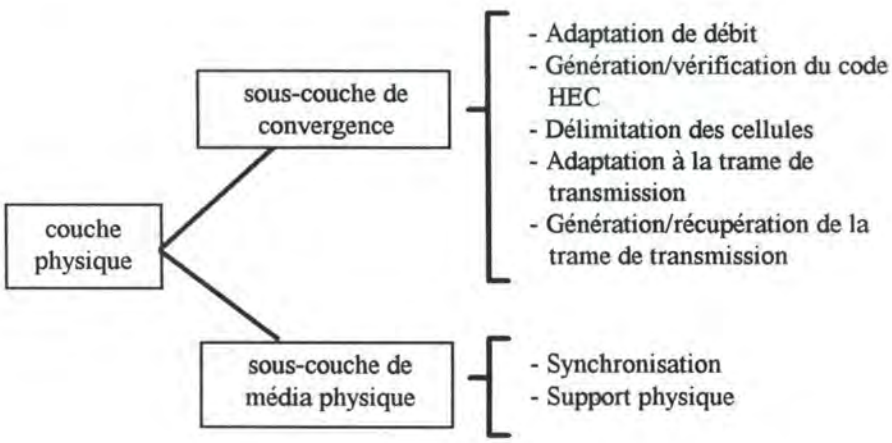


Figure 3-22 : Structure de la couche physique

3.4.1. La couche "média physique"

La sous-couche média physique constitue le niveau le plus bas dans le protocole ATM et inclut uniquement les fonctions dépendantes du support physique utilisé. Elle assure les fonctionnalités de transmission de bits, le codage/décodage et, si nécessaire, la translation électro-optique. Dans de nombreux cas, le support physique sera constitué par de la fibre optique. D'autres supports comme le câble coaxial ou la paire torsadée sont également utilisables.

Les fonctions de synchronisation regroupent la génération et la réception de signaux compatibles avec le support utilisé.

3.4.2. La couche "convergence au niveau transmission"

Parmi les fonctions remplies par la sous-couche de convergence au niveau transmission, celles de génération et d'adaptation à la trame de transmission dépendent du type d'interface utilisé entre l'utilisateur et le réseau. Les premiers paragraphes décrivent séparément ces deux fonctions pour les deux options d'interface disponibles (de type SDH ou orienté cellule). Les autres fonctions de la sous-couche de convergence peuvent, en principe, être mises en oeuvre de la même manière pour les deux types d'interface.

I. Génération et adaptation à la trame de transmission à l'interface de type SDH

La norme de transmission SDH (*Synchronous Digital Hierarchy*) fonctionne sur base d'une émission synchrone de trames de transmission. Ces trames, comme l'illustre la *Figure 3-23*, représentent un ensemble d'octets constitué de neuf lignes et 270 colonnes. Le flux de cellules ATM est projeté dans un conteneur constitué de neuf lignes et 260 colonnes. Les 10 colonnes initiales correspondent au surdébit nécessaire pour ce type d'interface. Les informations d'OAM (*Operation And Maintenance*) de la couche physique utilisent une partie de ce surdébit.

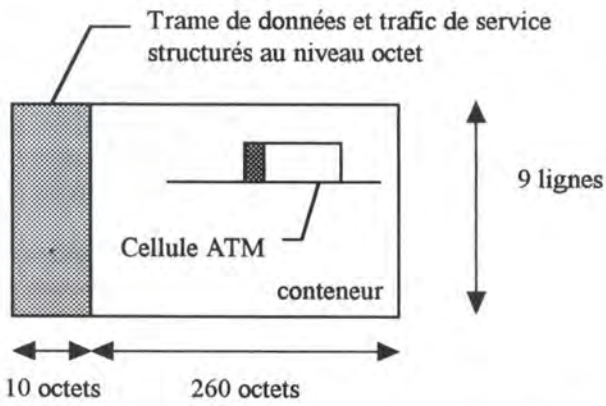


Figure 3-23 : Trame de transmission à l'interface UNI de type SDH

Cette norme de transmission SDH dispose de la souplesse nécessaire pour transporter des types de signaux très différents tels que des canaux RNIS ou des cellules ATM. Ainsi ce mode de transmission a été proposé comme structure d'interface pour le RNIS à large bande. Une telle implémentation d'interface usager-réseau offre l'avantage d'être complètement compatible avec celle utilisée entre noeuds du réseau. Cela évite la conversion, sinon inévitable, des signaux échangés entre des usagers à travers le réseau.

Cependant, l'interface usager-réseau de type SDH présente certains inconvénients. Un premier réside dans le fait que la capacité importante de surcharge réservée dans le SDH n'est pas réellement nécessaire au niveau de cette interface. Deuxièmement, la génération de la trame SDH qui est structurée au niveau octet, nécessite la prise en compte de fonctions qui ne sont pas nécessaires si l'interface est complètement orientée cellule.

Pour ces deux raisons, deux types d'interface ont été standardisés : l'un basé sur le mode de transmission synchrone SDH et l'autre sur le simple multiplexage de cellules. Bien que le débit de l'interface orientée cellule ne soit pas fondamentalement déterminé par une quelconque structure de trame mais qu'il puisse être choisi librement, il a été décidé au niveau de l'ITU-T, d'adopter le même débit pour les deux types d'interface afin de faciliter leur interopérabilité sur le réseau.

II. Génération et adaptation à la trame de transmission à l'interface de type cellule

Les interfaces orientées cellule, quel que soit leur débit (155 Mbit/s ou 622 Mbit/s) sont constituées d'un flux continu de cellules. Ce flux est illustré à la *Figure 3-24*. L'intervalle maximum entre deux cellules successives de la couche physique est de 26 cellules de la couche ATM. Donc, après 26 cellules ATM contiguës, une cellule de la couche physique est insérée afin d'adapter la capacité de transfert au débit de l'interface.

Quand aucune cellule de la couche ATM n'est disponible, des cellules de la couche physique sont insérées. Les cellules ainsi insérées du côté émetteur peuvent être soit des cellules vides, soit des cellules d'OAM (*Operation And Maintenance*) de la couche physique, en fonction des exigences d'exploitation, d'administration et de maintenance. Ces informations d'OAM sont véhiculées dans des cellules spécifiques de la couche physique, identifiées par des séquences de bits particulières qui leur sont exclusivement réservées dans l'en-tête de cellule.

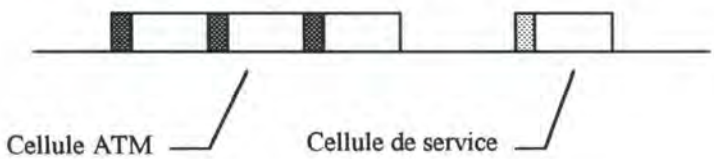


Figure 3-24 : Interface UNI orientée cellule

III. Adaptation de débit

Chaque fois qu'aucune cellule assignée, non assignée ou d'OAM n'est disponible pour la transmission, une cellule vide est insérée pour adapter le flux de cellules au débit de transmission. Toutes ces cellules supplémentaires sont éliminées du côté récepteur. L'adaptation au débit regroupe l'insertion et l'élimination des cellules vides.

Les cellules vides sont identifiées par une séquence standardisée dans l'en-tête de cellule

IV. Génération/vérification du code HEC

Selon le modèle de référence de protocole RNIS à large bande, le contrôle d'erreur sur l'en-tête des cellules ATM est une fonction de la couche physique. Chaque émetteur calcule la valeur du code de contrôle HEC (*Header Error Control*) à partir des quatre premiers octets de l'en-tête de la cellule et insère le résultat dans le cinquième octet (champ HEC). La valeur du code HEC est définie comme " le reste de la division modulo 2 par le polynôme générateur $x^8 + x^2 + x + 1$, du produit x^8 multiplié par le contenu de l'en-tête hors champ HEC".

Ce code HEC permet de :

- corriger des erreurs simples (affectant un seul bit),
- détecter certaines erreurs multiples (concernant plusieurs bits),

dans l'en-tête de cellule ATM. Ces deux fonctionnalités de traitement d'erreur sont assurées par les équipements récepteurs des cellules.

V. Délimitation des cellules

La "délimitation des cellules définit le processus qui permet l'identification des limites d'une cellule". Le mécanisme proposé par l'ITU-T pour la délimitation des cellules est basé sur la corrélation entre les bits d'en-tête à protéger et les bits de contrôle correspondants.

Dans l'état Recherche, est effectuée une vérification bit à bit du champ d'en-tête présumé. Quand les règles de codage du HEC semblent respectées, un en-tête est présumé avoir été identifié et le mécanisme passe à l'état Pré-sync (pré-synchronisation). A partir de ce moment, le contrôle de corrélation du HEC est effectué cellule par cellule. Si δ HEC's successifs valides ont été rencontrés, le système passe dans l'état Sync (synchronisation) ; sinon, il retourne à l'état Recherche. Le système passe de l'état Sync à l'état Recherche si α HEC's successifs invalides ont été détectés.

Avec $\alpha = 7$, un système ATM à 155 Mbit/s sera dans l'état Sync pendant plus d'une année même si le taux d'erreur sur les bits est d'environ 10^{-4}. Avec un $\delta = 6$, le même système, présentant le même taux d'erreur de bit, nécessitera environ 10 cellules ou 28 μs pour repasser dans l'état Sync après une perte de synchronisation au niveau cellule.

Chapitre 4

Spécifications de la partie ATM du logiciel Tcplp

Dans ce chapitre, nous allons présenter les problèmes inhérents à la compréhension des concepts de base du protocole ATM tels qu'exposés au chapitre précédent. Nous détaillerons ensuite pour chacun de ces concepts de base, la manière que nous avons choisie pour l'illustrer et le faire comprendre au mieux à l'étudiant. Les choix posés ont été réfléchis à la lumière des éléments théoriques dégagés au chapitre 2 sur le design des didacticiels.

La structure de ce chapitre est assez similaire à celle du chapitre précédent faisant état des concepts de base du protocole ATM.

4.1. Généralités

4.1.1. Principes généraux de conception

Le chapitre 2 énumère des principes directeurs utiles à suivre dans la conception d'un didacticiel qui se veut de qualité. Nous nous sommes efforcés tout au long de la conception de notre produit de privilégier l'interactivité en déléguant autant que possible l'initiative à l'élève.

I. Conditions préalables à la création d'un didacticiel

Nous avons établi le cahier des charges de notre produit avec Philippe van Bastelaer et Véronique Nachtergaele qui tous deux ont une expérience de l'enseignement dans le domaine des réseaux. De plus, en qualité d'étudiant, nous avons pu nous-même être confrontés aux difficultés du protocole ATM. Cette pluralité de points de vue dans la définition des finalités du produit respecte bien deux des quatre conditions préalables à la création d'un logiciel énumérées au paragraphe 2.2.1. En effet, les personnes précitées ont une expertise dans le domaine ATM et ont enseigné les contenus qui constitueront la base du logiciel. Ils sont donc familiers aux

difficultés rencontrées par ceux qui sont confrontés à l'apprentissage des contenus. L'interface a été développée sur base de règles reprises dans le cours d'interface Homme-Machine de Jean Vanderdonckt [JVD95] et grâce aux conseils prodigués par Daniel Muller sur les interfaces des didacticiels développés avec OSF/MOTIF.

Cependant, pour les deux conditions préalables restantes, nous ne les avons qu'à moitié respectées. La première d'entre elles imposait d'avoir du goût pour le travail en équipe ce qui était le cas mais ce travail d'équipe était limité aux seules spécifications. Pour le reste du développement du logiciel, nous avons procédé seul. Concernant la seconde condition qui imposait de ne pas apporter ses préventions ou croyances sur ce qu'on peut faire ou ne pas faire avec l'informatique, nous ne pouvions objectivement nous y tenir, compte tenu de notre formation d'informaticien.

II. Le rôle du didacticiel

Le rôle que nous avons voulu donner à notre didacticiel est un rôle d'aide visant à apprendre et à faire comprendre les concepts de base du protocole ATM. Ces concepts seront représentés graphiquement afin de faciliter leur apprentissage et leur compréhension.

III. Le choix des concepts-clés

Comme nous l'avions mentionné au chapitre 2, l'ordinateur peut jouer un rôle important dans la mise en évidence de concepts difficiles. Mais avant de penser à la manière idéale de représenter les concepts difficiles, il faut dégager ces concepts.

Nous avons tiré du protocole ATM quatre concepts jugés difficiles. Ces concepts sont les suivants :

- le caractère asynchrone de l'émission de cellules ATM qui seront transmises dans des cellules physiques synchrones.
- le rôle de la couche AAL et des différents services qu'elle fournit.
- le modèle bi-dimensionnel du modèle de référence du RNIS à large bande.
- la présence de connexions virtuelles à deux niveaux : VP et VC.

Les réponses que nous avons apportées à ces difficultés seront proposées dans la suite de ce chapitre.

IV. Le point de vue de l'élève

Nous avons décidé, pour rester cohérent avec le didacticiel TcpIp existant, de suivre un canevas de simulation temporelle. La manière d'appréhender le temps sera de le ralentir. En effet la circulation d'informations dans un réseau informatique quelconque se déroule en une fraction de temps trop courte pour être simulée en temps réel. Nous proposerons donc une vue ralentie du processus de communication dans un réseau ATM ainsi qu'une possibilité de faire un arrêt sur image pour que l'élève puisse analyser l'état du réseau à un moment donné.

Nous avons voulu que l'élève soit investi d'un rôle des plus actifs. Ses possibilités d'action seront détaillées dans la table des responsabilités ci-dessous.

La position spatiale que l'élève adoptera par rapport à la représentation que nous avons réalisée est celle d'un observateur se plaçant sous un angle de vue de 45 degrés au dessus du réseau. Ce réseau est formé de machines disposées en ellipse pour donner une perspective 3D (3-Dimensions) à notre graphisme. Cette perspective 3-D est conservée dans tout le didacticiel.

V. Le type de didacticiel

Nous avons déjà évoqué à plusieurs reprises le type de didacticiel que nous avons appliqué à notre produit. Il s'agit de la simulation du processus de transfert d'informations dans un réseau de type ATM. Ce logiciel peut aussi être qualifié de tutoriel dans la mesure où il dispense un enseignement.

VI. La table des responsabilités

Nous avons déjà évoqué le rôle actif que nous voulons faire jouer à l'élève dans notre didacticiel. Ce rôle se manifeste par les actions suivantes qu'il peut réaliser :

- Choisir deux machines pour établir une communication.
- Choisir une vue "grossière" d'une connexion entre les deux machines ou une vue détaillée de celle-ci.
- Choisir une couche du protocole ATM parmi les trois couches AAL, ATM et physique pour analyser son rôle au sein du protocole et voir la connexion logique à ce niveau entre des deux machines choisies.
- Lancer l'animation entre les deux machines choisies.
- Contrôler le déroulement de cette animation en l'interrompant à tout moment et en décidant de la reprise à l'état initial ou l'état suspendu.
- Contrôler la vitesse d'exécution de l'animation.
- Voir le format des PDU's de la couche sélectionnée et leur formation à partir des PDU's des couches supérieures.
- Voir la table de routage utilisée à chaque noeud du réseau ATM et la manière dont elle est utilisée.

Le rôle de l'enseignant reste minimal dans notre produit. Il peut se limiter à expliquer les fonctionnalités du logiciel et donner un schéma type d'utilisation du logiciel de manière à profiter pleinement de l'enseignement dispensé.

Le rôle de l'ordinateur, quant à lui, demeure celui de l'exécutant au service de l'élève.

4.1.2. Un protocole asynchrone

La caractéristique asynchrone du protocole ATM est celle qui en fait sa particularité. Cependant, elle est assez facile à comprendre. Nous avons représenté l'asynchronisme au niveau de chaque couche. L'asynchronisme signifie que les cellules sont émises en fonction des besoins. Ces besoins sont négociés au niveau AAL en choisissant un type de service que l'on veut que le réseau remplisse. En fonction de ce service désiré, le débit sera variable ou constant. S'il est constant, les cellules sont générées de manière synchrone alors que si le service ne requiert pas un débit constant, les cellules sont générées de manière purement asynchrone.

4.1.3. La cellule

La cellule est l'élément de base du protocole ATM. Une cellule est constituée d'un en-tête de 5 octets et d'un champ d'informations de 48 octets. Le champ d'informations correspond au AAL-PDU. Au niveau d'une connexion logique entre les couches AAL des deux machines sélectionnées, ce ne sont que des AAL-PDU's qui sont échangés.

Au niveau ATM, ce sont des cellules qui sont échangées du point de vue logique. Nous avons voulu montrer la différence entre ces deux niveaux logiques de transmission (couche AAL et couche ATM) par la mise en évidence de l'en-tête de cellule au niveau ATM. Cet en-tête marque la caractéristique de la couche ATM. Nous avons choisi de le mettre en évidence en utilisant une couleur différente de celle utilisée pour représenter le champ d'informations. Pour garder la cohérence entre les couches, la couleur qui sera utilisée au niveau AAL pour représenter le champ d'informations sera également utilisée dans la cellule de la couche ATM pour représenter le même champ.

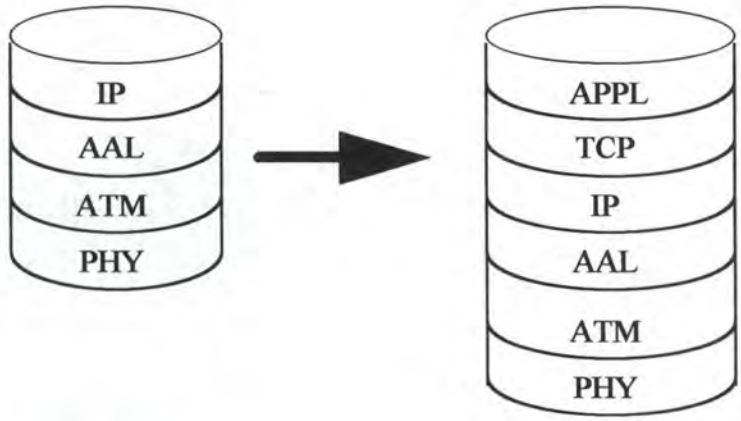
Au niveau physique, ce ne sont que des bits qui sont transmis et la différence entre en-tête et champ d'informations n'a plus de raison d'être. Cependant, tous les bits provenant d'une même cellule sont représentés comme un bloc distinct des autres blocs symbolisant d'autres cellules. Nous avons choisi de représenter ces blocs de la même couleur que celle employée pour le champ d'informations dans les autres couches. Cette identité de couleur a été préservée pour montrer la différence entre les cellules vides et celles porteuses d'informations. Nous aurions pu représenter ces blocs de bits avec une taille légèrement supérieure à celle du champ d'informations de la couche AAL mais, à échelle proportionnelle, la différence n'aurait pas été perceptible à l'écran. Nous avons alors préféré insister sur la différence entre ces deux présentations identiques de choses différentes au niveau de la vue détaillée du format des PDU's dans la fenêtre prévue à cet effet.

4.1.4. La découpe en couches

La découpe en couches que nous avons présentée dans le modèle de référence du RNIS à large bande est un point difficile à comprendre par les étudiants. Cette difficulté est en partie due au changement du modèle de référence habituellement utilisé qui est le modèle OSI. De plus, ce modèle du RNIS est décomposé en plans qui n'existent pas dans le modèle OSI. Comme la découpe en couches est différente dans les deux modèles, l'élève a des difficultés à rattacher les fonctions traditionnelles des couches OSI aux couches du modèle du RNIS.

Nous avons choisi de ne représenter que le plan d'usager tel qu'il est défini dans le modèle de référence, sans nous occuper des plans de contrôle et de gestion. Nous expliquons donc les trois couches du protocole ATM ainsi que les fonctions majeures remplies par chacune d'elle.

Chaque machine appartenant à un réseau ATM est représentée avec un nombre différent de couches selon qu'il s'agit d'un hôte simple, d'une passerelle ou d'un noeud ATM. Nous avions d'abord pensé représenter les hôtes simples comme un cylindre de quatre couches: PHY (Physique), ATM, AAL et IP (*Figure 4-1(a)*). Mais pour veiller à la cohérence avec ce que TcpIp possédait déjà, nous avons ajouté deux couches supplémentaires. Il s'agit des couches TCP et APPL (Application) comme illustré à la *Figure 4-1(b)*.

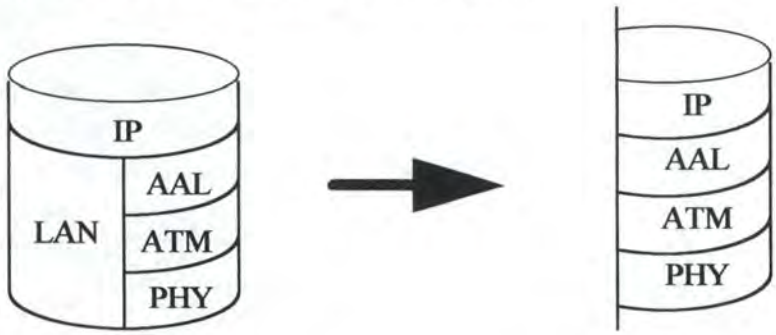


(a) hôte ATM 1ère idée

(b) hôte ATM définitif

Figure 4-1 : Représentation d'un hôte simple d'un réseau ATM

La représentation initiale à laquelle nous avions songé pour les passerelles est illustrée à la *Figure 4-2(a)*. Nous voulions montrer le lien que les passerelles établissent avec le réseau "voisin" par la division du cylindre en deux parties ; chacune supportant un protocole (dans l'exemple de la *Figure 4-2(a)*, le réseau auquel le réseau ATM est relié par la passerelle est un LAN). Finalement, nous avons opté pour le dessin d'un cylindre coupé en deux dans le sens vertical mais dont une des deux parties seulement est visible (*Figure 4-2(b)*). Ce choix rend la représentation d'un passerelle ATM cohérente avec les passerelles existantes dans TcpIp.



(a) Passerelle entre un réseau ATM et un LAN

(b) Passerelle au sein d'un réseau ATM

Figure 4-2 : Représentation d'une passerelle dans un réseau ATM

Les noeuds ATM sont les éléments chargés du routage dans un réseau ATM. Ils ne sont composés que des deux couches Physique (PHY) et ATM. Leur représentation ne pose donc aucun problème (*Figure 4-3*).

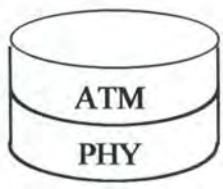


Figure 4-3 : Représentation d'un noeud ATM

4.1.5. Le passage des PDU's

Le logiciel TcpIp consacre une place non négligeable aux formats des paquets échangés au sein de l'Internet. Ce format peut être demandé à tout moment par l'utilisateur en allant choisir l'item de menu "*Datagram format*" dans le menu "*Monitor*". Les formats proposés correspondent au niveau physique du sous-réseau par lequel le datagramme IP passe au moment où le format est demandé. Par contre, le format des PDU's des différentes couches par lesquelles l'information passe entre la couche IP et la couche physique n'est pas abordé. Il en est de même en ce qui concerne le passage des PDU's entre les couches d'une même machine.

Dans la partie concernant ATM, nous avons décidé de prendre en compte le passage des PDU's entre les couches d'une même machine vu l'aspect plus détaillé de la simulation.

L'utilisateur peut choisir, lorsqu'il se trouve dans la vue détaillée d'un réseau ATM, une couche du protocole ATM pour analyser ses fonctions dans l'ensemble du protocole. Une fois choisie, il décide quand il le veut de démarrer l'animation. Celle-ci commence au niveau IP quelle que soit la couche choisie. Dans la fenêtre dédiée au format des PDU's nous affichons d'abord le format d'un datagramme IP. A partir de là, nous avons choisi de modifier l'optique choisie dans TcpIp qui était d'afficher les formats successifs les uns après les autres dans la même fenêtre et chaque fois en remplacement du précédent. Ce que nous voulons, c'est montrer à l'utilisateur les stades successifs de transformation d'un datagramme IP pour aboutir à la génération des cellules.

Ce premier changement en a induit un autre pour permettre une bonne compréhension du processus de passage des PDU's. Toutes les étapes se déroulent dans une même fenêtre. Cette fenêtre change de taille dynamiquement en fonction du stade de décomposition atteint par le datagramme IP. Ce datagramme reste représenté pendant toute la construction des cellules et les PDU's des couches inférieures à IP s'affichent en dessous de celui-ci.

L'évolution des étapes dans cette fenêtre prévue pour le format des PDU's se fera en parallèle à la simulation qui se passe dans la fenêtre du sous-réseau.

I. Couche AAL

La couche AAL se situe juste en dessous de la couche IP et le passage d'un datagramme IP à cette couche va être examiné en tenant compte des deux sous-couches formant la couche AAL. Nous avons choisi de représenter deux grandes étapes entre la couche IP et la couche AAL.

La première correspond au passage du datagramme IP à la sous-couche de convergence (CS). Le datagramme IP est d'abord représenté avec ses champs constitutifs et lorsqu'il est transmis, on ne le représentera plus que comme un bloc. Ses champs n'ont d'ailleurs de signification qu'au niveau IP et pas aux niveaux inférieurs. Au niveau CS, sont ajoutés un en-tête (*CS Header*) et une terminaison (*CS Trailer*). Ces deux phases (passage du datagramme et ajout des champs de contrôle de protocole) sont illustrées à la *Figure 4-4 (I)*.

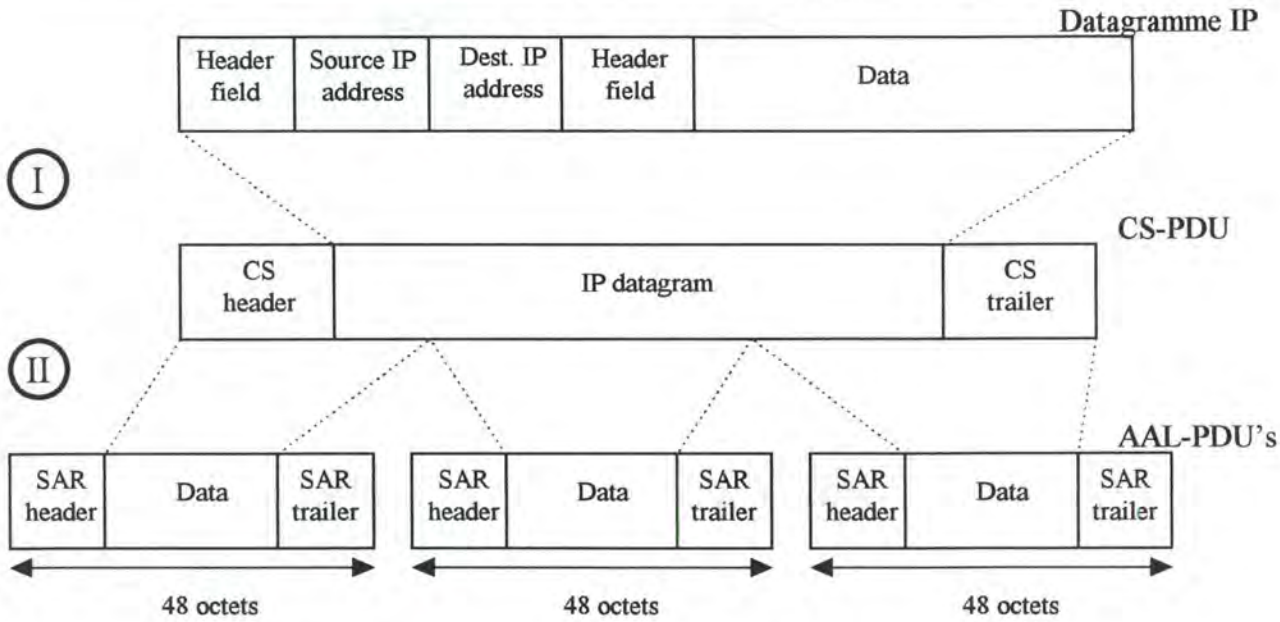


Figure 4-4 : Transformation du datagramme IP en AAL-PDU's

La seconde grande phase constitue le passage de la sous-couche CS à la sous-couche de segmentation et réassemblage (SAR). Comme la *Figure 4-4 (II)* le montre, le CS-PDU est divisé en blocs d'une longueur inférieure ou égale à 48 octets. A ces blocs sont ajoutés au niveau de la sous-couche SAR, des en-têtes (*SAR Header*) et des terminaisons (*SAR Trailer*). Les SAR-PDU's correspondent aux AAL-PDU's. Leur longueur fixe, caractéristique d'ATM, est indiquée en dessous des PDU's. Le nombre de AAL-PDU's que nous avons choisi de représenter n'est motivé que par la place disponible sur l'écran d'ordinateur et pour garantir la lisibilité des champs des PDU's.

Lorsque la construction des AAL-PDU's est terminée, ceux-ci sont envoyés via un lien virtuel à la couche AAL de la machine choisie en tant que réceptrice. Au fur et à mesure que les cellules sont envoyées, elles disparaissent de la fenêtre dédiée au format des PDU's.

Une fois les trois AAL-PDU's envoyés, nous avons décidé de placer dans la fenêtre propre au format des PDU's, le format détaillé du CS-PDU et du SAR-PDU (= AAL-PDU). Par format détaillé on entend le format reprenant les champs avec leur signification et leur longueur. Ces formats dépendent uniquement du type de service choisi par l'utilisateur lors de la négociation du service[3]. Les champs réellement présents dans ces PDU's ne sont pas toujours tous représentés. En effet, nous n'avons pris en compte que ceux qui sont en rapport avec les fonctions que nous avons choisi de représenter. Les autres champs sont réunis sous le vocable *Header* ou *Trailer* selon qu'ils sont situés au début ou à la fin du PDU.

Les formats que nous avons adoptés pour l'AAL de type 1 sont schématisés à la *Figure 4-5*. Le format du CS-PDU pour le protocole AAL de type 1 que nous avons retenu, se réduit au minimum. Il s'agit des champs *CS header, CS trailer* et du champ d'informations qui comprend le datagramme IP. Pour le SAR-PDU, nous avons négligé la découpe plus fine des deux champs d'en-tête. En outre, nous avons conservé le champ *SAR Trailer* qui est vide dans le cas de l'AAL de type 1. Pour symboliser cela, nous l'avons différencié grâce à un format distinct du reste du PDU.

[3] Cette négociation sera détaillée dans les fonctions de la couche AAL.

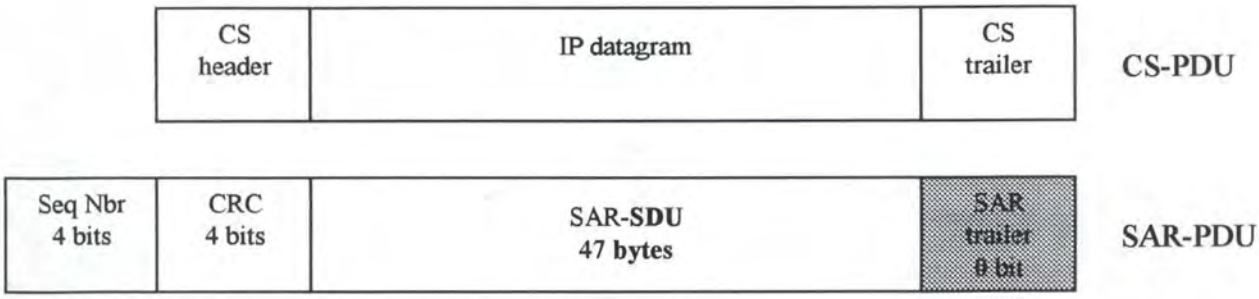


Figure 4-5 : Format des CS-PDU et SAR-PDU pour l'AAL de type 1

Pour l'AAL de type 3/4, le CS-PDU possède huit champs dont la plupart ne nous intéressent pas. Nous n'avons conservé que le champ *data* qui contient le datagramme IP et le champ *Padding* utilisé pour cadrer la terminaison du PDU sur une longueur multiple de 32 bits. On comprend facilement son rôle lorsqu'on voit que ce CS-PDU est divisé en un nombre entier de blocs de 44 bytes dans le cas de l'AAL de type 3/4.

Par contre, nous avons schématisé tous les champs du SAR-PDU avec leur longueur. Ces deux PDU's sont illustrés à la *Figure 4-6*.

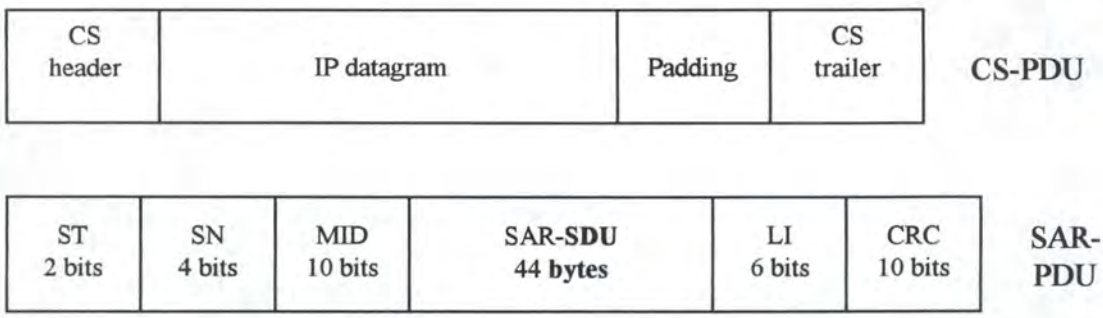


Figure 4-6 : Format des CS-PDU et SAR-PDU pour l'AAL de type 3/4

Le dernier type d'AAL dont nous parlerons est l'AAL de type 5. C'est en principe ce dernier qui est le plus approprié lorsque ATM est utilisé sous IP. Le CS-PDU correspondant à ce type d'AAL est pratiquement représenté dans notre logiciel tel qu'il existe réellement. Nous avons seulement regroupé les champs UU (*CPCS user-to-user indication*) et CPI (*Common Part Indicator*) sous l'appellation "*Trailer fields*".

Le SAR-PDU n'est constitué que de son champ d'informations qui occupe les 48 bytes disponibles pour l'ensemble du SAR-PDU. L'absence d'informations supplémentaires est justifiée par la vocation de ce type d'AAL à être simple et efficace. Nous avons gardé dans notre dessin les champs *SAR Header* et *SAR Trailer* qui sont différenciés par un format spécial.

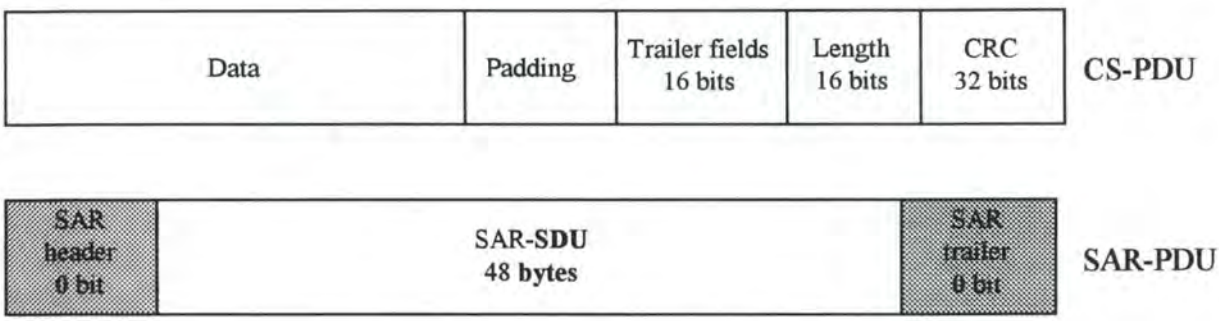


Figure 4-7 : Format des CS-PDU et SAR-PDU pour l'AAL de type 5

II. Couche ATM

La couche ATM se situe juste en dessous de la couche AAL. Elle reçoit les AAL-PDU's de 48 octets. Ceux-ci viennent s'insérer dans le champ d'informations des cellules générées au niveau ATM.

Il est évident que les PDU's de la couche ATM se basent sur les PDU's de la couche AAL. Cependant, comme nous avons voulu permettre à l'utilisateur de se concentrer sur une couche pour en étudier les caractéristiques, il ne nous a pas paru opportun de reprendre au niveau ATM, toute la construction des AAL-PDU's. Pour cette raison, le datagramme IP présent au début sera remplacé par l'AAL-PDU dès que l'animation passera de la couche IP à la couche AAL.

Les étapes de construction de cellules que nous avons établies sont les suivantes :

1. Remplacement du datagramme IP par l'AAL-PDU's.
2. Prise en compte du AAL-PDU comme le champ d'informations de la cellule.
3. Ajout des champs d'en-tête au champ d'informations.

Elles sont schématisées à la *Figure 4-8* ci-dessous par leur numéro. Nous avons décidé d'indiquer la longueur des champs de la cellule en dessous de leur nom.

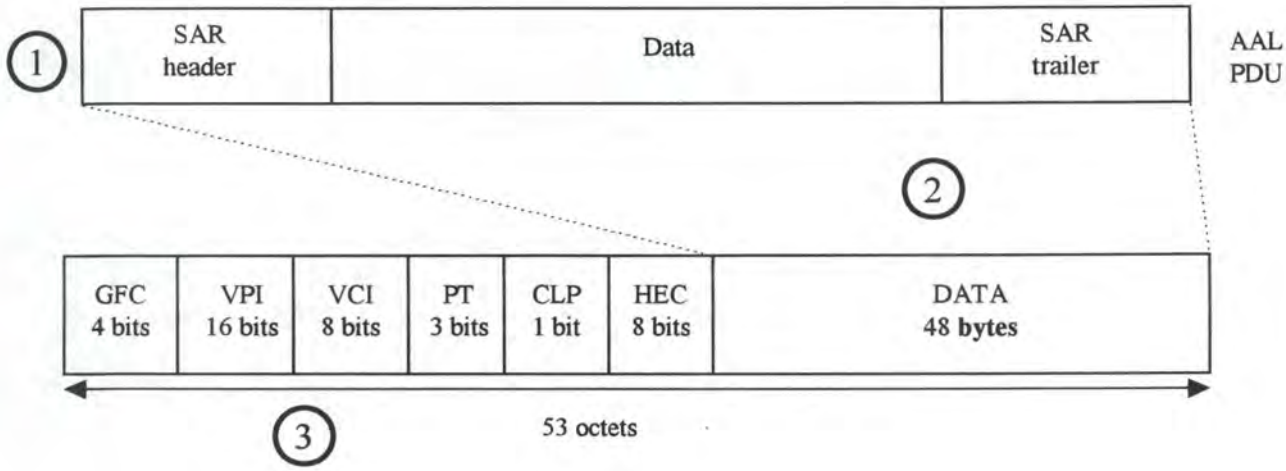


Figure 4-8 : Formation d'une cellule à partir d'un AAL-PDU

De même qu'au niveau de la couche AAL, les cellules quittent la machine d'origine une fois qu'elles sont "prêtes". Au moment où la première cellule entre sur le réseau, le contenu de la fenêtre des PDU's change pour ne plus contenir que le format de la cellule complète. La couche ATM s'occupe du routage au sein du réseau en plus de la génération des en-têtes de cellules. Ce routage s'effectue à chaque noeud du réseau où sont changés les VCI et VPI et par conséquent le HEC qui est calculé sur les autres champs de l'en-tête. Ces autres champs peuvent également être modifiés en fonction notamment de la charge du réseau. Par conséquent, nous avons choisi de mettre en évidence ce changement d'en-tête par des couleurs différentes.

D'autre part, le format de l'en-tête des cellules est différent à l'interface UNI et à l'interface NNI. Ce changement se passe au premier noeud du réseau.

III. Couche physique

La couche physique est la plus basse dans la structure en couche et elle reçoit les PDU's de la couche ATM. Elle n'a aucune mission de modification des PDU's qu'elle reçoit. Elle ne fait que les traduire en flots de bits et les adapter au support physique. Le terme cellule est aussi utilisé au niveau physique et nous avons voulu garder le même format que celui employé au niveau ATM même si la découpe en champs n'a plus de sens ici. Les champs restent à titre indicatif.

Nous venons de présenter la manière dont nous comptons représenter les caractéristiques de base d'un réseau ATM. Maintenant il nous reste à montrer comment nous schématisons un réseau ATM. Mais avant cela, nous allons expliquer brièvement les raisons qui nous ont poussés à développer une nouvelle interface.

4.1.6. La nouvelle interface de TcpIp

L'interface initiale de TcpIp comprenait trois types de réseaux plus une ligne louée. Cette configuration ne nous permettait pas d'ajouter facilement un nouveau type de réseau et surtout de pouvoir faire face à de possibles nouveaux types venant s'ajouter plus tard. En outre, cette configuration était rigide pour l'utilisateur. Nous avons donc décidé d'établir une configuration

type dans laquelle chaque réseau pourrait supporter n'importe quel protocole. L'utilisateur définirait l'Internet à sa manière. L'interface que nous proposons est celle représentée à la *Figure 4-9* mais elle n'est malheureusement pas encore implémentée. L'utilisateur devrait pouvoir attribuer n'importe quel protocole à chacun de ces quatre réseaux.

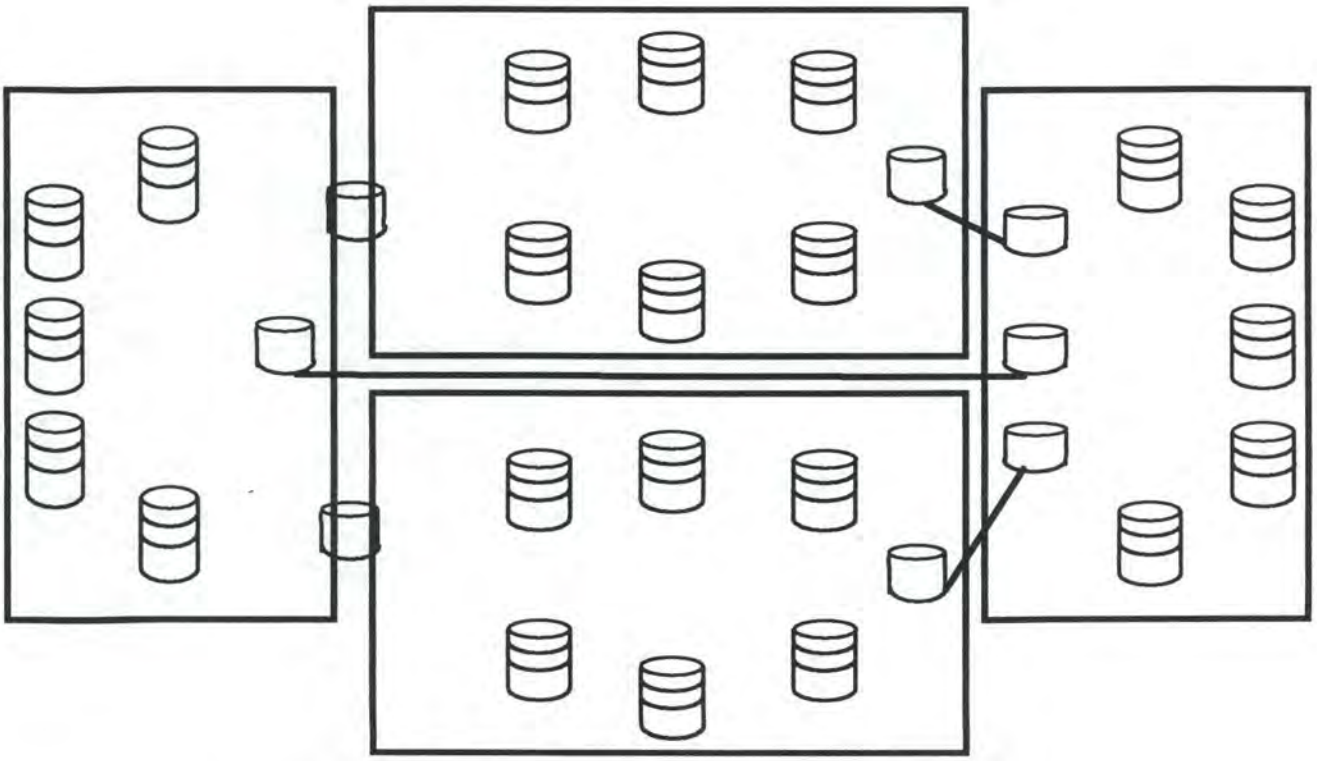


Figure 4-9 : Nouvelle interface du didacticiel TcpIp

4.1.7. Les représentations d'un réseau ATM

La représentation du réseau ATM, ou plutôt des réseaux ATM, va être conforme aux configurations qui apparaissent sur le dessin de la *Figure 4-9*. Il y a donc trois configurations différentes : deux configurations verticales correspondant aux deux réseaux verticaux de la *Figure 4-9* et une configuration horizontale correspondant aux réseaux horizontaux de cette même figure.

Dans la fenêtre du sous-réseau, les configurations sont basées sur celles présentées dans la fenêtre principale mais de manière plus détaillée. Les noeuds ATM et les lignes physiques sont représentés en plus des machines et des passerelles. Nous avions d'abord placé 11 noeuds dans chaque réseau : un par machine et trois supplémentaires au milieu. Finalement nous n'avons conservé que les trois noeuds centraux étant donné la taille limitée de la fenêtre graphique contenant la représentation du sous-réseau (*Figure 4-10*). De plus les trois noeuds sont amplement suffisants pour illustrer les mécanismes de routage.

Cette fenêtre de sous-réseau correspond à la notion d'écran-clé dont nous avons parlé au chapitre 2. Cet écran constitue le coeur de notre produit à partir duquel l'élève explore les options offertes et matérialisées par les menus. Le dialogue entre l'application et l'utilisateur se

réalise dans la zone de commentaires prévue dans le fond de la fenêtre. Un bouton de commande est placé dans le coin inférieur droit de la fenêtre. Cette position permet un déplacement minimal de la vue de l'utilisateur.

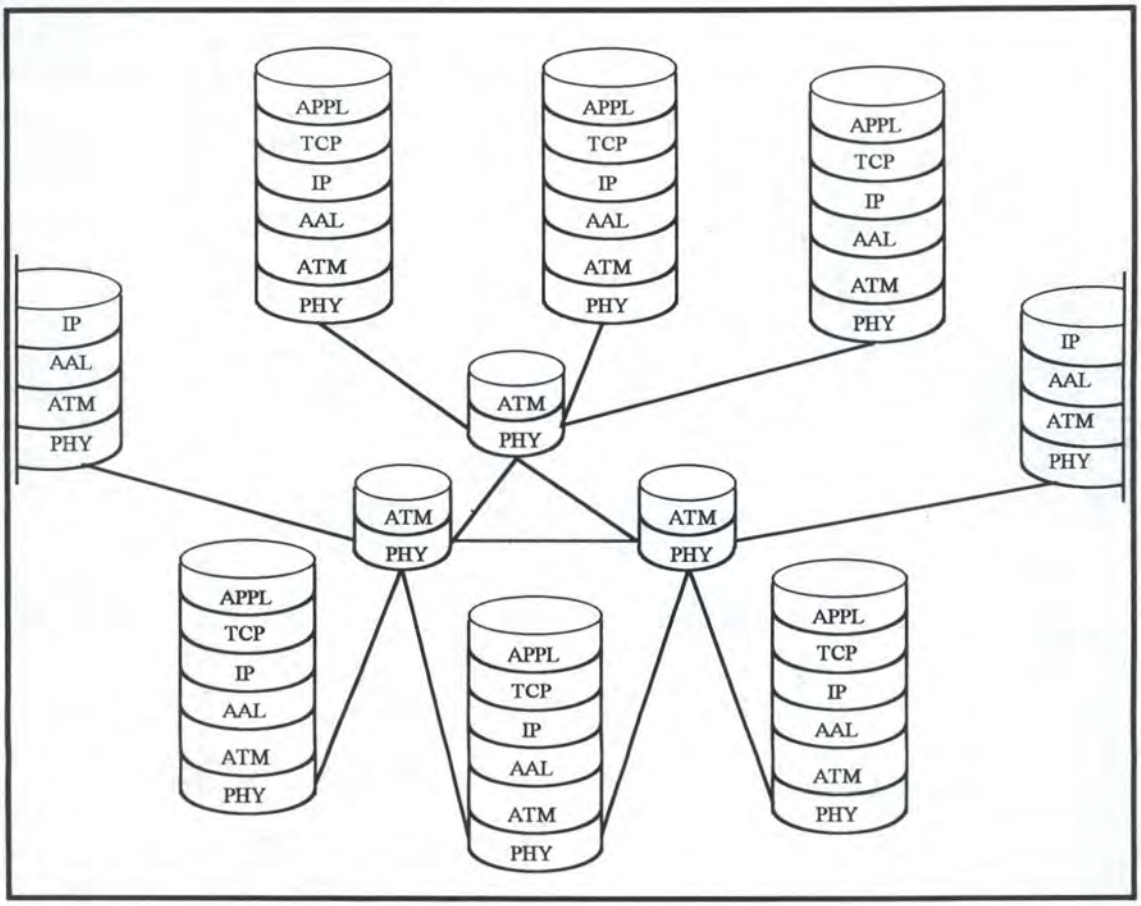


Figure 4-10 : Représentation de la configuration horizontale du réseau ATM

4.2. Couche AAL

La couche AAL remplit plusieurs fonctions dans le protocole ATM. Celles-ci ont été détaillées dans le chapitre précédent. Il s'agit, d'une part, de réaliser la convergence entre les couches supérieures et la couche ATM et, d'autre part, de segmenter les PDU's des couches supérieures pour les adapter à la longueur des cellules. Ces deux fonctions sont remplies par les deux sous-couches formant la couche AAL. Nous les décrivons toutes deux dans notre logiciel pour éclaircir au maximum les difficultés de compréhension associées à la couche AAL et à son rôle dans le protocole ATM. Nous avons déjà accordé une grande place aux fonctions de cette couche dans la section décrivant le passage des PDU's.

4.2.1. Une connexion virtuelle

Les couches AAL des deux machines sélectionnées communiquent entre elles par l'échange d'AAL-PDU's. Cet échange est réalisé par la création d'une liaison virtuelle entre les

deux couches AAL concernées. Une seconde[2] connexion virtuelle est établie entre deux autres machines. Un autre rôle de cette seconde connexion est d'illustrer le fait que plusieurs connexions virtuelles peuvent exister simultanément. Nous avons représenté cette connexion au niveau AAL pour rester cohérent avec les autres niveaux où elle aura davantage d'utilité. Une connexion virtuelle entre deux couches AAL est représentée à la *Figure 4-11*. On remarque que l'effet 3-D a été gardé par rapport à la position que nous donnons à l'élève dans l'environnement.

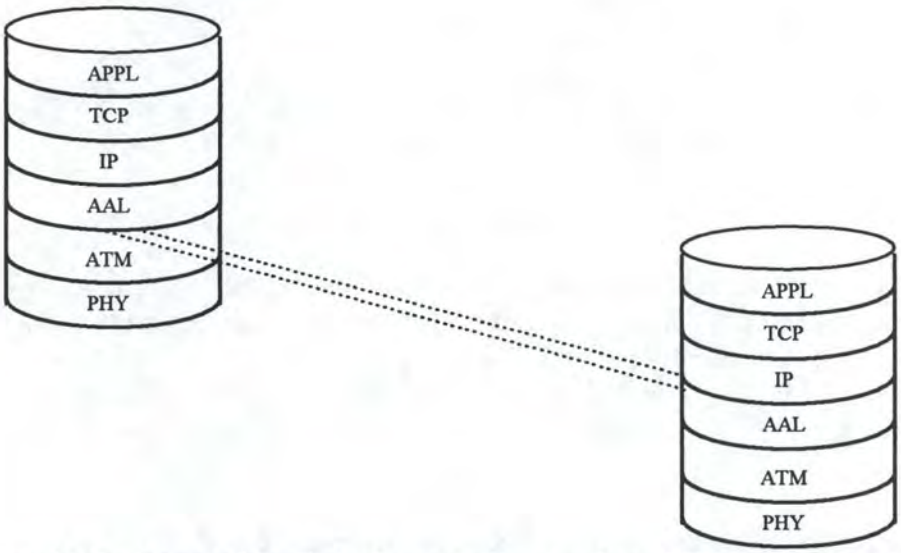


Figure 4-11 : Connexion virtuelle entre les couches AAL de deux machines

4.2.2. Négociation du service

Nous proposons à l'utilisateur de négocier le service qu'il désire. Pour cela, nous lui proposons un choix parmi trois types de protocole AAL. Nous avons choisi de n'exposer que les types 1, 3/4 et 5. Le type 2 n'étant pas encore bien défini, nous ne l'avons pas intégré.

Chaque type de protocole AAL répond à un service déterminé. Nous avons voulu proposer à l'utilisateur un récapitulatif des caractéristiques de chaque type d'AAL ainsi qu'un exemple de cas dans lequel chaque type était utilisé. Voici les caractéristiques que nous avons choisies par type d'AAL.

- AAL de type 1
  - Trafic à taux constant.
  - Synchronisation entre la source et la destination.
  - Orienté connexion.
  - *exemple* : transmission de la voix.

- AAL de type 3/4
  - Trafic à taux variable.
  - Pas de synchronisation entre la source et la destination.
  - Orienté connexion ou sans connexion préalable.
  - *exemple* : transmission de données.

[2] Il faut entendre seconde en tant que secondaire parce que cette "seconde" connexion apparaît avant l'autre.

- AAL de type 5
  - Trafic "efficace" à taux variable.
  - Mêmes services que la couche 3/4 mais avec un surdébit plus faible.
  - Protocole plus simple et plus efficace que le protocole 3/4.
  - *exemple* : IP sur ATM.

Le choix effectué par l'utilisateur va entraîner plusieurs conséquences. La plus visible est sans doute l'émission synchrone des AAL-PDU's pour le type 1 alors que l'émission est asynchrone pour les deux autres types.

La seconde conséquence touche le format des AAL-PDU's. Ils ont déjà été décrits ci-dessus et nous n'y reviendrons pas ici.

4.2.3. L'enchaînement des actions

Un échange d'informations entre deux entités AAL situées sur des machines distinctes se déroule en plusieurs étapes. Chacune de ces étapes correspond à l'accomplissement d'une action par une des couches concernées par la connexion. Il peut s'agir de la couche IP ou de la couche AAL tant du coté émetteur que du coté récepteur. Nous avons établi un scénario type pour la communication entre deux couches AAL. Ce scénario est décomposé en plusieurs étapes dont voici la liste :

1 Formation du datagramme IP au niveau de la couche IP.
2 Passage du datagramme IP à la couche AAL avec l'adresse IP de destination.
3 Choix de la machine de destination dans le réseau ATM en fonction de l'adresse IP de destination.
4 Apparition d'un lien virtuel entre les couches AAL des machines source et destination.
5 Réalisation de la fonction de convergence dans la machine émettrice par l'adjonction de champs au CS-SDU qui est en fait le datagramme IP.
6 Réalisation de la fonction de segmentation du CS-PDU dans la machine émettrice
7 Ajout, dans la machine émettrice, des champs de contrôle de protocole en en-tête et en terminaison des SAR-SDU's.
8 Envoi des SAR-PDU's formés.
9 Transfert des données.
10 Réception des SAR-PDU's au niveau de la couche AAL réceptrice.
11 Effacement du lien virtuel.
12 Réassemblage du CS-PDU au niveau de la sous-couche SAR du récepteur par extraction du contenu informationnel des SAR-PDU's.
13 Extraction du contenu informationnel du CS-PDU (égal au datagramme IP) au niveau de la couche CS de la machine réceptrice.
14 Passage du datagramme IP à la couche IP côté récepteur.
15 Livraison de l'information contenue dans le datagramme IP par la couche IP de la machine réceptrice.

4.3. Couche ATM

L'analyse de cette couche est semblable à celle présentée pour la couche AAL. Un scénario d'enchaînement des actions a été établi. Mais avant de passer à la description de ce scénario, nous allons passer en revue les fonctions qui sont du ressort de la couche ATM et expliquer pourquoi nous avons choisi de les introduire dans notre logiciel et comment nous les avons représentées. Détaillons d'abord la manière d'établir une connexion virtuelle.

4.3.1. Une connexion virtuelle

Pour analyser les fonctions de la couche ATM dans le protocole, nous nous intéressons à l'échange d'informations entre les couches ATM des deux machines sélectionnées. Cet échange passe par les noeuds du réseau. Une connexion virtuelle est donc établie entre les couches ATM des deux machines par l'intermédiaire des couches ATM de un ou deux noeuds. Le nombre de noeuds intervenant dans une connexion est toujours minimal.

Une seconde[3] connexion virtuelle est créée entre deux couches ATM de deux machines lorsque l'utilisateur désire analyser les fonctions de la couche ATM. Cette connexion[4] est créée de telle manière à avoir toujours une liaison virtuelle en commun avec la première connexion. Si cette première connexion n'est composée que de deux liaisons et donc d'un seul noeud, la seconde connexion reliera n'importe quelle machine différente des deux machines sélectionnées à la machine de destination de la première connexion (Cfr. *Figure 4-12*).

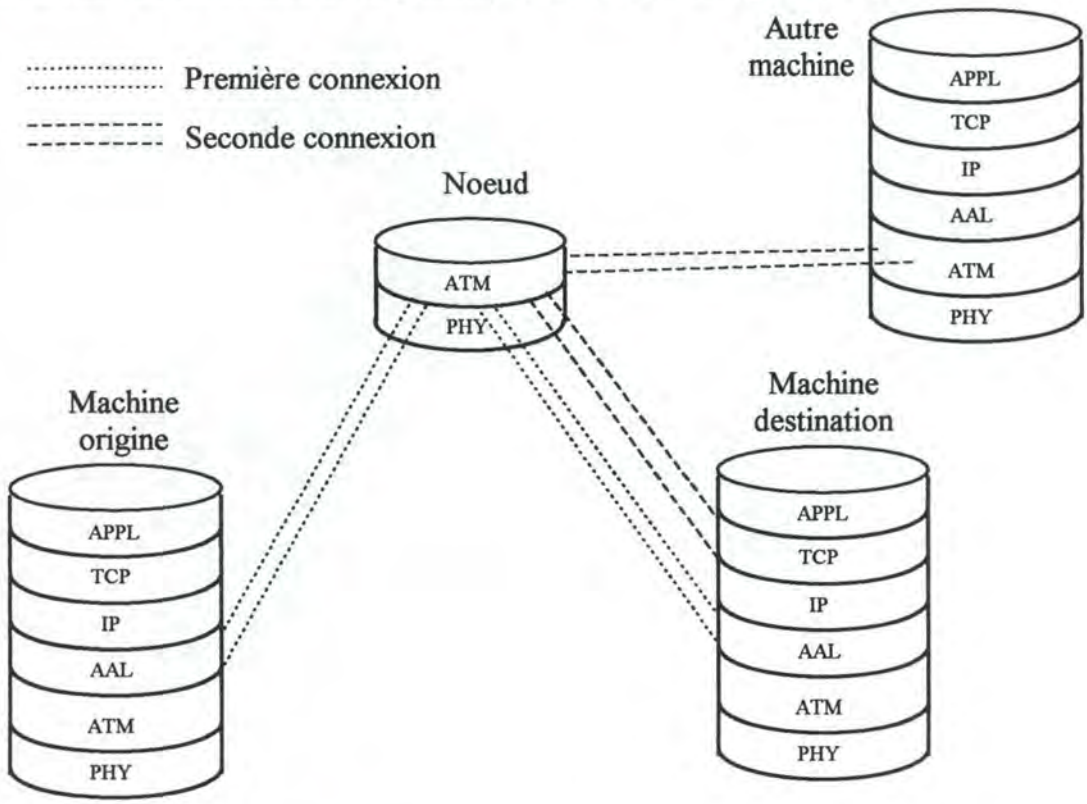


Figure 4-12 : Connexions virtuelles couches ATM dans un réseau ATM

[3] Cfr. Supra (note 2)
[4] Rappelons qu'une connexion virtuelle est composée de liaisons virtuelles.

Si la première connexion contient deux noeuds, la liaison virtuelle établie entre ces deux noeuds est celle choisie pour être la liaison commune aux deux connexions virtuelles.

4.3.2. Les fonctions de la couche ATM

I. Contrôle de flux

La fonction de contrôle de flux n'est pas encore mise en oeuvre dans les configurations actuelles de réseaux ATM. Pour cette raison, nous ne la représentons pas dans notre didacticiel.

II. Traduction des VCI/VPI

La seconde fonction de la couche ATM est celle de routage des cellules. Pour ce faire, les couches ATM des noeuds du réseau changent le couple VPI-VCI. Ce couple identifie chaque connexion entrant dans un noeud et chaque cellule passant sur une connexion possède son couple identifiant VPI/VCI. Cette traduction se fait après consultation d'une table de routage. Nous avons donc choisi de modéliser une table de routage pour montrer à l'utilisateur la manière dont cette opération de traduction se réalise. Dans un réseau ATM, chaque noeud possède sa propre table de routage qu'il met à jour chaque fois qu'une liaison de VC ou de VP est établie ou supprimée. Pour réduire le nombre de tables de routage, nous avons choisi de représenter uniquement les tables des machines ou noeuds concernés par la communication qui nous occupe.

Nous utilisons une méta-table des tables de routage pour gagner de la place à l'écran. Cette considération de spécification est imprégnée de notre connaissance de l'informatique et de ses limitations (Cfr. Chapitre 2 §2.1). La méta-table que nous avons conçue est représentée à la *Figure 4-13* ci-dessous.

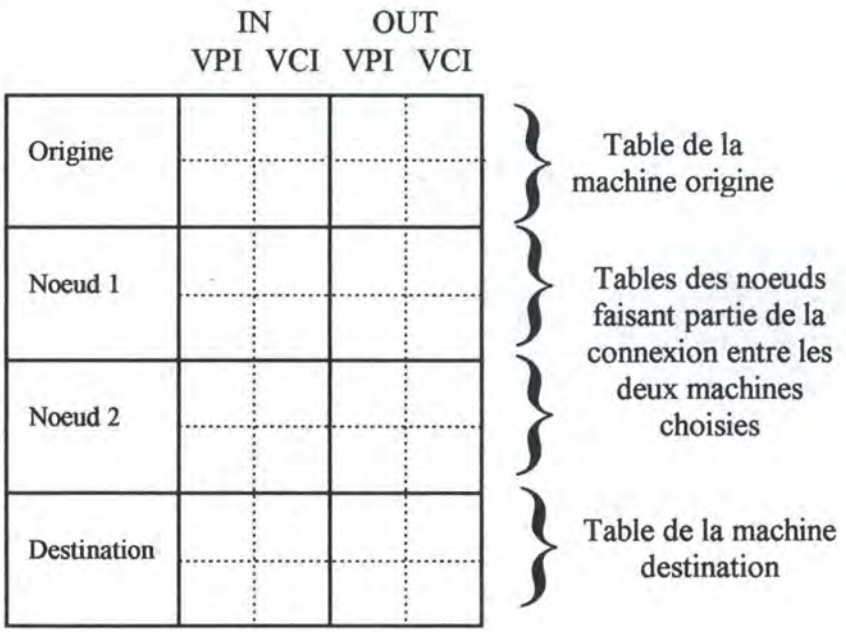


Figure 4-13 : Table des tables de routage pour un réseau ATM

Cette table des tables est suffisante pour tout cas de figure envisageable dans le réseau que nous avons construit et tel qu'il est représenté à la *Figure 4-10*. Elle reprend, pour chaque élément du réseau inclus dans la connexion entre les deux machines choisies, les couples VPI-VCI des liaisons virtuelles formant la connexion.

Les tables de routage possèdent les propriétés suivantes :

- La table de la machine d'origine ne possédera jamais qu'un seul couple VPI/VCI en sortie (OUT).
- La table de la machine de destination ne possédera jamais de couples en sortie. Elle contiendra un ou deux couples en entrée d'après le nombre de connexions virtuelles arrivant à la machine de destination.
- La table du noeud 2 sera vide si la connexion ne passe que par un seul noeud du réseau.
- La table du noeud 1 recensera toujours deux liaisons en entrée (IN) et deux en sortie (OUT).

III. Multiplexage et démultiplexage des cellules

La troisième fonction dont la couche ATM est responsable est celle de multiplexage et démultiplexage. Cette fonction est illustrée au niveau ATM mais est également présente au niveau physique. C'est notamment pour cette fonction de multiplexage qu'une seconde connexion a été créée. Les cellules qui arrivent au noeud possédant deux liaisons virtuelles en sortie sont orientées vers la "bonne" liaison. Par "bonne" liaison il faut entendre la liaison dont le couple VPI/VCI identifiant, se retrouve dans l'en-tête des cellules.

IV. Génération et extraction des en-têtes de cellules

La dernière fonction que nous avons prise en compte pour la couche ATM est celle de génération et d'extraction des en-têtes de cellules. Nous ne pouvons qu'illustrer cela dans la fenêtre réservée aux PDU's. Du côté émetteur, nous ajoutons l'en-tête au champ d'informations provenant de la couche AAL. Du côté récepteur par contre, cet en-tête est supprimé pour obtenir la partie informative et la transmettre à la couche AAL. La fonction d'extraction est réalisée au niveau des couches ATM des noeuds du réseau. Ils doivent extraire l'en-tête des cellules pour analyser la valeur du couple VPI/VCI et générer, après consultation dans la table de routage, un nouvel en-tête qui est ajouté au champ d'informations restant inchangé.

Le seul champ qui ne soit pas généré au niveau de la couche ATM est le champ de contrôle d'erreur HEC. Ce champ est généré au niveau de la couche physique sur base des bits résultant de la traduction en signaux physiques des champs de l'en-tête hors HEC. Pour marquer cette particularité, nous dessinons ce champ d'une manière différente dans le format de la cellule.

4.3.3. L'enchaînement des actions

De la même manière que nous avions procédé pour la couche AAL, nous avons établi un scénario d'enchaînement des actions effectuées dans la machine émettrice, la machine destinatrice mais aussi dans les noeuds du réseau. Le scénario que nous avons adopté est décrit ci-après.

1 Formation du datagramme IP au niveau de la couche IP.
2 Passage du datagramme IP à la couche AAL avec l'adresse IP de destination.
3 Réalisation des fonctions de la couche AAL.
4 Demande d'ouverture de connexion au niveau ATM par la couche AAL.
5 Création d'une liaison virtuelle entre la couche ATM de la machine émettrice et la couche ATM du noeud d'entrée dans le réseau.
6 Mise à jour des tables de routage du noeud d'entrée et de la machine émettrice.
7 Si nécessaire (si la machine destinatrice n'a pas le même noeud d'entrée que la machine émettrice):
  7.1 Création d'une liaison virtuelle entre la couche ATM du noeud d'entrée de la machine émettrice et celle du noeud d'entrée de la machine destinatrice.
  7.2 Mise à jour de la table de routage dans les deux noeuds concernés.

8 Création d'une liaison virtuelle entre la couche ATM du noeud d'entrée de la machine destinatrice et la couche ATM de cette même machine destinatrice.
9 Mise à jour de la table de routage de la machine destinatrice et son noeud d'entrée.
10 Du côté de la machine émettrice :
  10.1 Passage des AAL-PDU's de la couche AAL à la couche ATM.
  10.2 Ajout des en-têtes de cellules avec des valeurs significatives dans chaque champ sauf dans le champ HEC qui a sa valeur au niveau physique.
  10.3 Envoi des cellules.

11 Transfert des données.
12 Dans la couche ATM du noeud d'entrée de la machine origine :
  12.1 Réception des cellules.
  12.2 Consultation de la table de routage.
  12.3 Changement de l'en-tête des cellules.
  12.4 Envoi des cellules.

13 Transfert des données.
14 Si nécessaire (si la machine destinatrice n'a pas le même noeud d'entrée que la machine émettrice): au niveau de la couche ATM du second noeud dans la connexion :
  14.1 Réception des cellules.
  14.2 Consultation de la table de routage.
  14.3 Changement de l'en-tête des cellules.
  14.4 Envoi des cellules.

15 Transfert des données.
16 Du côté de la machine réceptrice :
  16.1 Réception des cellules.
  16.2 Suppression de la connexion virtuelle.
  16.3 Suppression de l'en-tête des cellules.
  16.4 Passage des parties informatives des cellules de la couche ATM à la couche AAL.
  16.5 Réalisation des fonctions de la couche AAL.
  16.6 Passage du datagramme IP à la couche IP.
  16.7 Livraison de l'information contenue dans le datagramme par la couche IP

4.4. Couche PHYSIQUE

Passons finalement à la couche physique. Nous avons procédé de la même manière que pour les deux couches supérieures. Parmi l'ensemble des fonctions lui incombant, nous n'en avons retenu que quatre. Nous n'avons pas pris en considération les fonctions de la sous-couche de média physique parce que celles-ci sont trop liées au support physique. La fonction de délimitation des cellules a elle aussi été écartée parce qu'elle n'est effectuée que très rarement. En effet, dans un système ATM à 155,520 Mbit/s, si on décide d'appeler cette fonction une fois que 7 codes HEC successifs sont invalides, la fonction sera appelée moins d'une fois par an en moyenne en supposant que le taux d'erreur sur les bits est d'environ 10^{-4}.

4.4.1. Les fonctions de la couche ATM

Les fonctions que nous avons voulu développer sont celles de génération du code HEC, de génération/récupération de la trame, de génération et d'adaptation à la trame de transmission et d'adaptation du débit.

I. Génération/vérification du code HEC

La génération du code de contrôle d'erreur (HEC) est une opération basée sur les bits formant les premiers champs des cellules hors champ HEC. Or dans notre simulation nous ne parlons jamais de bits à quelque endroit que ce soit. Nous mentionnons alors textuellement la génération du code HEC et parallèlement, nous annulons la différence graphique que possédait le champ HEC dans la représentation du format de la cellule par rapport aux autres champs de l'en-tête. Ce changement graphique se passe lorsque la couche ATM transmet les cellules à la couche physique pour être émises.

La vérification de ce code d'erreur HEC est mentionnée textuellement. En outre, nous considérons que ce code est toujours correct ce qui n'entraîne aucun traitement d'erreurs.

II. Génération et adaptation à la trame de transmission

Les fonctions de génération et d'adaptation à la trame de transmission dépendent du type d'interface utilisé entre l'utilisateur et le réseau. Nous avons choisi de représenter l'interface orientée cellule parce c'est la manière qui semble la plus naturelle dans un protocole orienté cellule. La fonction de génération est effectuée par la création d'un flux continu de cellules vides. L'adaptation à la trame de transmission est des plus simples avec une génération de trames orientées cellule. Chaque cellule prête est envoyée directement. Les cellules vides se distinguent des cellules non vides par une couleur distincte neutre.

III. Adaptation du débit

Chaque fois qu'aucune cellule assignée, non assignée ou d'OAM n'est disponible pour la transmission, une cellule vide est insérée pour adapter le flux de cellules au débit de transmission. Toutes ces cellules supplémentaires sont éliminées du côté récepteur. L'adaptation du débit a lieu également aux noeuds du réseau. La deuxième connexion que nous avons introduite dans le réseau intervient ici au niveau physique pour illustrer cette notion d'adaptation au débit. Sur la liaison commune, les flots de cellules venant de chaque connexion sont entrelacés de manière à respecter le débit de chaque connexion.

4.4.2. L'enchaînement des actions

Nous allons décrire ici comme nous l'avons fait pour les deux autres couches, le scénario d'enchaînement des actions effectuées dans la machine émettrice, la machine destinatrice mais aussi dans les noeuds du réseau. Le scénario que nous avons adopté est décrit ci-après.

1 Au niveau de la machine émettrice :
- 1.1 Formation du datagramme IP au niveau de la couche IP.
- 1.2 Passage du datagramme IP à la couche AAL avec l'adresse IP de destination.
- 1.3 Réalisation des fonctions de la couche AAL.
- 1.4 Passage des AAL-PDU's à la couche ATM.
- 1.5 Ajout de l'en-tête des cellules.
- 1.6 Passage des cellules à la couche physique.
- 1.7 Calcul du code de contrôle sur l'en-tête HEC.
- 1.8 Envoi des cellules.

2 Transfert des données.

3 Au niveau du noeud d'entrée de la machine d'origine :
- 3.1 Réception des cellules à la couche physique.
- 3.2 Vérification du code HEC.
- 3.3 Passage des cellules à la couche ATM.
- 3.4 Changement de l'en-tête des cellules en fonction des informations contenues dans les tables de routage.
- 3.5 Passage des cellules à la couche physique.
- 3.6 Calcul du code de contrôle HEC sur le nouvel en-tête.
- 3.7 Envoi des cellules.

4 Transfert des données.

5 Si le chemin physique contient un deuxième noeud, les mêmes opérations que celles menées de **3.1 à 3.7** sont réalisées au niveau de ce second noeud.

6 Au niveau de la machine de destination :
- 6.1 Réception des cellules à la couche physique.
- 6.2 Vérification du code HEC de chaque cellule.
- 6.3 Passage des cellules à la couche ATM.
- 6.4 Réalisation des fonctions de la couche ATM.
- 6.5 Passage des ATM-SDU's à la couche AAL.
- 6.6 Réalisation des fonctions de la couche AAL.
- 6.7 Passage du datagramme IP à la couche IP.
- 6.8 Livraison de l'information contenue dans le datagramme par la couche IP.

Chapitre 5

Conception de la partie ATM du didacticiel TcpIp

Le chapitre que nous proposons ici constitue la partie la plus conséquente du travail que nous avons réalisé sur le didacticiel TcpIp. La conception de toute la partie ATM du didacticiel est décrite dans ce chapitre ainsi que les modifications que nous avons apportées avant de commencer la partie ATM. Ces modifications concernent l'architecture logicielle et l'interface du didacticiel.

Nous décrirons dans un premier temps, les fonctionnalités de la partie ATM du didacticiel. Ensuite nous illustrerons les interfaces que nous avons conçues sur base des choix établis au chapitre précédent. Enfin, nous passerons en revue l'architecture logicielle que nous avons développée.

5.1. Les fonctionnalités du didacticiel

Le protocole que nous avons décrit dans notre travail s'insère dans le didacticiel TcpIp comme un nouveau protocole de sous-réseau à intégrer dans l'Internet. Comme nous l'avons déjà évoqué précédemment, la description du protocole ATM est beaucoup plus détaillée que celle des autres protocoles. Pour cette raison, la fenêtre contenant la vue détaillée des sous-réseaux est différente pour le sous-réseau ATM. Cette fenêtre est semblable à la fenêtre principale : elle contient des menus, une zone de commandes dans le fond formée d'une zone de commentaires, d'une échelle pour contrôler la vitesse de l'animation et d'un bouton de commande.

Les fonctionnalités que nous avons ajoutées sont dès lors liées à cette fenêtre. Les cinq premières fonctionnalités sont liées aux items de menu et les trois dernières à la zone de commandes.

5.1.1. Le choix d'une des couches du protocole ATM

Nous avons voulu séparer les couches du protocole ATM pour pouvoir analyser de manière approfondie les actions dont chacune est responsable. Avant toute animation, l'utilisateur doit choisir une couche pour laquelle l'animation va être effectuée. Cette sélection se fait en choisissant un des trois items du menu Animate. Ces trois items sont illustrés à la *Figure 5-1*. L'ordre de leur présentation symbolise la hiérarchie des trois couches AAL, ATM et physique. Nous avons ajouté des accélérateurs de menu qui eux aussi rappellent l'ordre des couches.

AAL layer	Alt+3
ATM layer	Alt+2
PHYsical layer	Alt+1

Figure 5-1 : Menu Animate

Ces items de menu ne sont "activables" que lorsque deux machines ont été sélectionnées dans la fenêtre principale représentant l'Internet. La sélection d'un niveau d'animation provoque alors plusieurs effets. D'abord, le bouton de commande[5] permettant de contrôler l'animation, qui était "inactivable", devient "activable". Deuxièmement, quel que soit le niveau choisi, une deuxième connexion est créée de manière à montrer l'activité du réseau dans son ensemble et à illustrer certaines fonctions des couches comme le multiplexage. Cette deuxième connexion n'a aucune influence au niveau AAL. La troisième conséquence est provoquée au cours de la sélection de l'item "PHYsical layer" qui déclenche l'envoi de cellules vides sur tous les liens physiques du réseau pour mettre en évidence le principe d'émission synchrone des cellules.

5.1.2. La demande d'un moniteur pour le datagramme

La seconde fonctionnalité et la troisième correspondent aux items du menu Monitor. Ce menu est illustré à la *Figure 5-2*. La fonctionnalité de demande d'un moniteur de datagramme associée à l'item de menu "Datagram's format" est déjà présente dans la fenêtre principale de TcpIp. Elle permet à l'utilisateur de voir le format des PDU's formés à partir du datagramme IP. Cet item de menu a été repris de la fenêtre principale parce que nous considérons que, lorsque l'utilisateur décide de voir ce qui se passe dans le réseau ATM, il ne s'intéresse plus à ce qui se passe dans la fenêtre principale.

Datagram's format	Alt+D
Routing tables	Alt+R

Figure 5-2 : Menu Monitor

[5] Ce bouton correspond à une fonctionnalité qui sera expliquée plus loin.

La sélection de cet item de menu provoque les mêmes effets que la sélection de l'item de menu dans la fenêtre principale correspondant à la demande d'un moniteur pour le datagramme. L'effet du déclenchement de cette fonctionnalité est l'apparition à l'écran d'une nouvelle fenêtre dans laquelle se trouve une représentation graphique de l'état de modification du datagramme IP en PDU's de la couche choisie. Si aucune couche n'est encore choisie ou que l'animation n'a pas encore commencé, c'est la représentation du datagramme IP seul qui apparaît dans cette fenêtre.

Le contenu de cette fenêtre change dynamiquement parallèlement à l'évolution de l'animation dans le sous-réseau.

5.1.3. La demande de la vue de la table de routage

La fonctionnalité de demande de la vue de la table de routage correspond au deuxième item du menu Monitor présenté à la *Figure 5-2*. Cette fonctionnalité est matérialisée par un bouton on/off permettant de faire apparaître ou disparaître la table de routage.

Lorsque l'item est désactivé et que l'utilisateur l'active, deux effets peuvent en résulter. Si le niveau de l'animation choisi est celui correspondant à la couche ATM, une nouvelle fenêtre incluant la table de routage apparaît. Cette table contient seulement les valeurs des VPI/VCI propres à la deuxième connexion si l'animation n'est pas encore commencée. Si l'animation est déjà commencée, la table contient les couples VPI/VCI correspondant aux liaisons virtuelles déjà ouvertes.

Au contraire, si aucun niveau d'animation n'est encore choisi ou que le niveau n'est pas celui correspondant à ATM, une fenêtre d'information apparaît. Cette fenêtre signale à l'utilisateur que la table de routage n'apparaîtra que si l'animation se déroule au niveau ATM. En effet, la fonction de routage est reprise dans les attributions de la couche ATM et nous avons choisi d'illustrer cette fonction au niveau ATM uniquement bien qu'elle soit présente au niveau physique aussi.

Lorsque l'item est activé et que l'utilisateur le désactive, la fenêtre contenant la table de routage disparaît si elle était présente. Si elle n'était pas présente (parce qu'aucun niveau d'animation n'est sélectionné ou que le niveau sélectionné n'est pas ATM) elle n'apparaîtra pas si l'utilisateur choisi ATM comme niveau pour l'animation.

5.1.4. Retour à la fenêtre principale

Il nous reste à décrire la fonctionnalité correspondant au premier menu de notre fenêtre du sous-réseau ATM. Ce menu ne contient qu'un item de menu qui permet à l'utilisateur de revenir à la fenêtre principale en fermant la fenêtre reprenant la représentation détaillée du sous-réseau.

5.1.5. Affichage de la fenêtre logo

Les fonctions d'aide ne sont pas encore développées dans le didacticiel TcpIp. C'est pourquoi le menu Help ne contient qu'un item permettant d'afficher la fenêtre logo de l'application. Cette fenêtre contient les informations sur la création du logiciel.

5.1.6. Envoi des cellules et gestion de l'animation

La fenêtre du sous-réseau ATM nécessite le choix d'un niveau d'animation correspondant à une couche du protocole ATM avant de voir le déroulement de la communication entre les deux machines sélectionnées par l'utilisateur. Ce choix d'un niveau a comme effet premier, comme nous l'avons plus haut, le changement d'état du bouton de commande. Ce bouton *send* permet la gestion du déplacement des cellules. L'utilisateur démarre l'animation en appuyant sur ce bouton. Il peut de la même manière arrêter l'animation et la reprendre.

5.1.7. La modification de la vitesse de l'animation

La fonctionnalité de modification de la vitesse de l'animation est également reprise de la fenêtre principale. Elle se fait par l'intermédiaire d'une échelle graduée de 0 à 100. L'utilisateur règle la vitesse comme il le souhaite en déplaçant le curseur de cette échelle. Ce changement de vitesse s'applique directement à l'animation en cours ou aux animations à venir si aucune animation n'est commencée.

La variation de la vitesse sur cette échelle entraîne une modification automatique de l'échelle dans la fenêtre principale aussi pour montrer que les deux échelles sont liées et ont toujours la même valeur sauf pour la valeur minimale. Une modification de la vitesse dans la fenêtre principale a le même effet dans la fenêtre du sous-réseau.

5.1.8. La demande de l'animation step by step

La différence entre les deux échelles est située au niveau de leur borne inférieure. L'échelle de la fenêtre principale est graduée à partir de 1 alors que la nouvelle échelle dans le sous-réseau est graduée à partir de 0. Cette dernière valeur permet à l'utilisateur de voir une animation à vitesse nulle ce qui signifie qu'après chaque étape, l'animation s'arrête. Pour passer à l'étape suivante, l'utilisateur doit appuyer chaque fois sur le bouton *send* de gestion de l'animation. Cette méthode permet à l'utilisateur de décomposer et d'analyser une communication en détail à quelque niveau que ce soit.

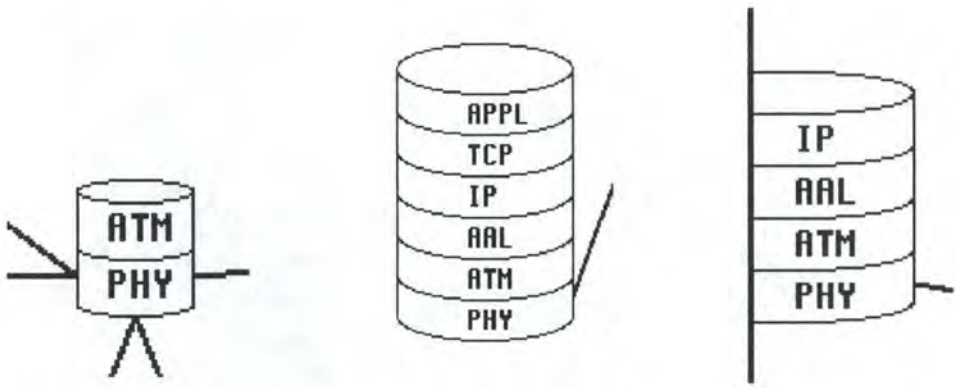
Nous venons de présenter les possibilités offertes à l'utilisateur pour mener son étude du protocole ATM. Ces fonctionnalités s'ajoutent à celles déjà présentes dans le didacticiel TcpIp et décrites dans [XGO95]. Il nous reste à décrire l'interface Homme-Machine et l'architecture logicielle.

5.2. Choix d'une interface

Cette partie est consacrée à la description de l'interface Homme-Machine que nous avons choisie pour mettre en oeuvre les choix établis au chapitre précédent sur les spécifications de la partie ATM. Nous allons décrire les fenêtres de l'interface en donnant les règles sur lesquelles nous nous sommes basés pour concevoir de telles fenêtres.

5.2.1. Les composants du réseau et le réseau

Nous allons montrer dans cette partie la manière dont nous avons représenté les éléments constituant le réseau ATM. La *Figure 5-3(a)* présente la représentation d'un noeud. La représentation d'une machine simple est montrée à la *Figure 5-3(b)* tandis que la représentation d'une passerelle ATM est illustrée à la *Figure 5-3(c)*.



(a) noeud ATM **(b)** machine simple ATM **(c)** passerelle ATM

Figure 5-3 : Représentation des éléments d'un réseau ATM

A partir de ces éléments, nous avons construit trois topologies de réseaux. La représentation du réseau horizontal est illustrée à la *Figure 5-4*.

Nous signalons que toutes les copies d'écran qui vont suivre ont été adaptées de manière à illustrer le mieux possible ce que nous avons représenté dans notre logiciel. En effet, nous avons utilisé la couleur pour différencier plusieurs notions et ces effets auraient été annulés à l'impression de la copie de l'écran si nous n'avions pas fait certaines modifications.

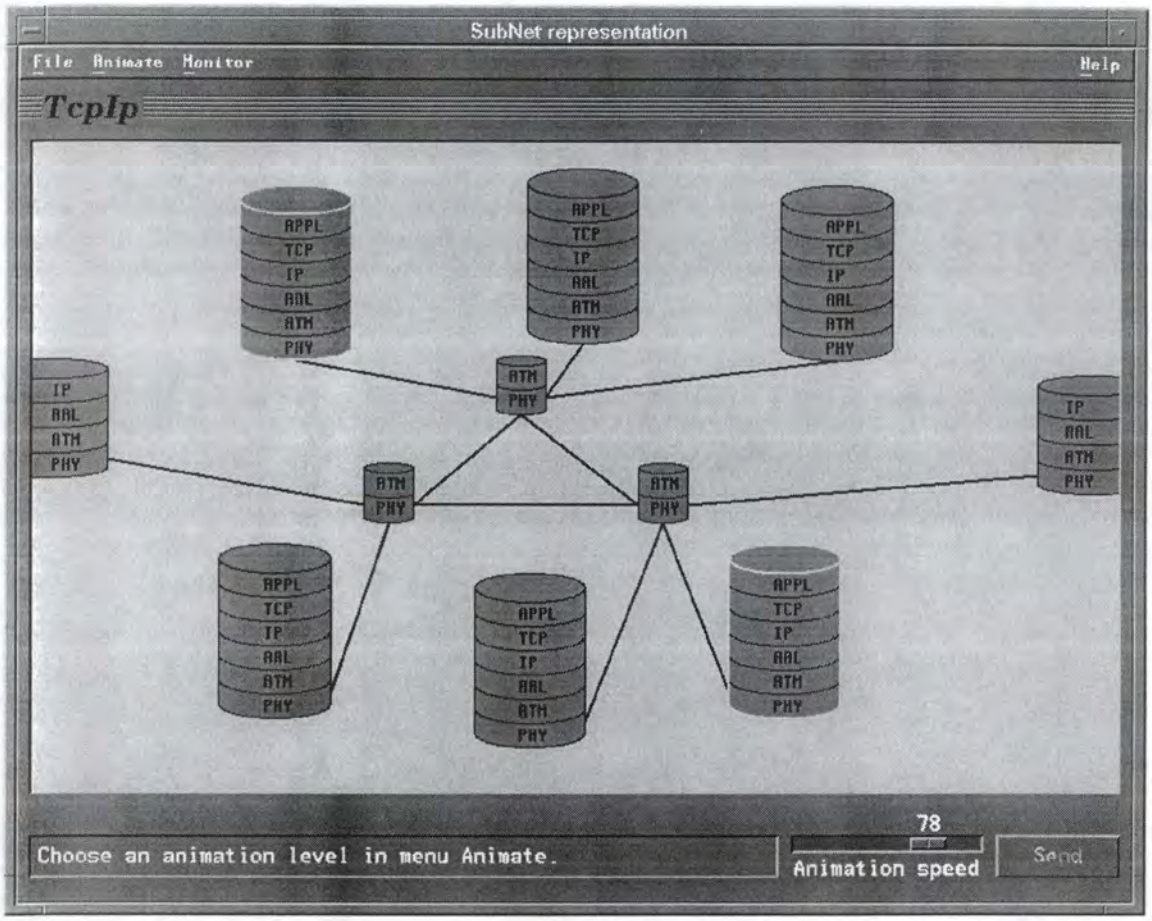


Figure 5-4 : Représentation du réseau ATM dans sa configuration horizontale

Ces trois topologies sont nécessaires pour la nouvelle interface de TcpIp. Nous avons voulu que la vue détaillée du sous-réseau garde la forme de la topologie utilisée dans la fenêtre principale. Chaque topologie contient 8 machines et 3 noeuds. Parmi ces 8 machines, 5 sont de simples machines et 3 sont des passerelles dans les réseaux à topologie verticale tandis que dans le réseau à topologie horizontale, il n'y a que 2 passerelles et 6 machines simples comme on peut le voir à la *Figure 5-4*.

Cette *Figure 5-4* montre également la manière dont la zone de commande est disposée. La zone de commentaires est suivie de l'échelle et du bouton de commande à droite de l'écran.

La *Figure 5-5* illustre la configuration verticale "de gauche". La configuration que nous avons utilisée "à droite" lui est symétrique. Les termes "gauche" et "droite" sont utilisés ici par rapport à la vue qu'a l'utilisateur lorsque la fenêtre de l'Internet est affichée.

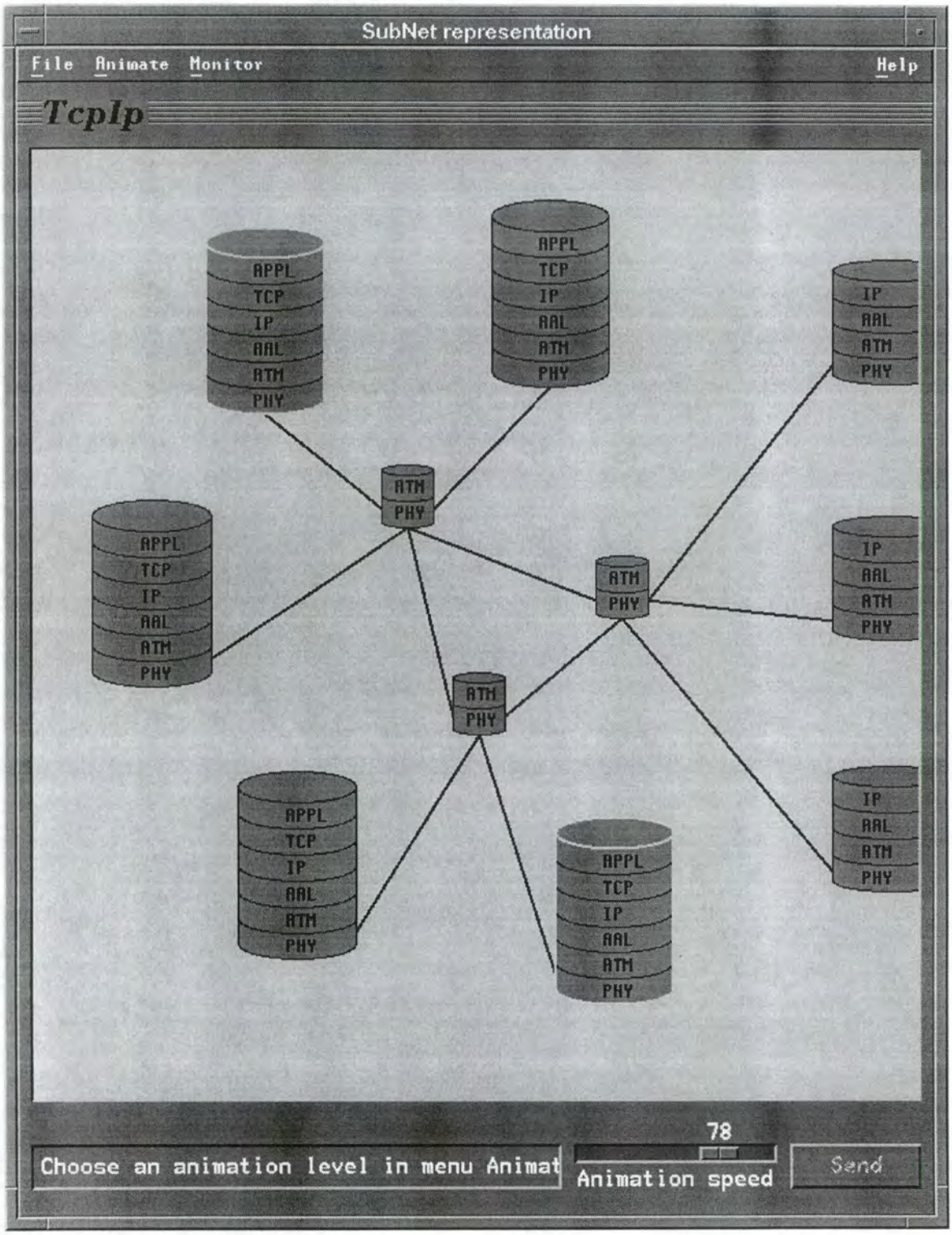


Figure 5-5 : Représentation du réseau ATM dans sa configuration verticale "gauche"

5.2.2. La couche AAL

L'utilisateur peut choisir la couche AAL comme sujet d'analyse. Il choisira donc l'item de menu "*AAL layer*" dans le menu "*Animate*". La première chose qui se passe avant toute animation est la négociation du service. Cette négociation est illustrée à la *Figure 5-6*.

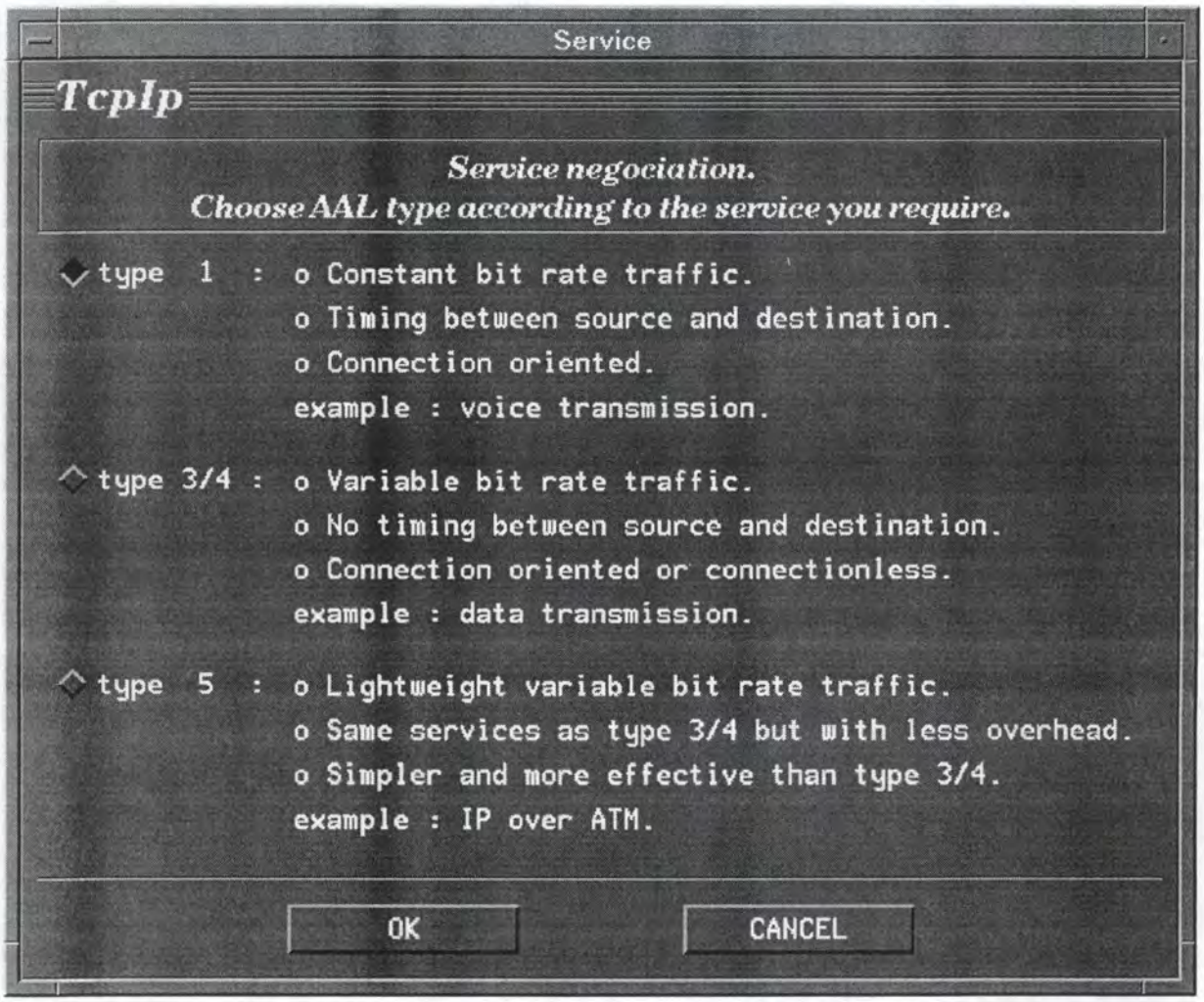


Figure 5-6 : Négociation du service au niveau de la couche AAL

L'utilisateur peut donc choisir le service qu'il souhaite en fonction des caractéristiques de chaque protocole AAL qui lui sont rappelées.

La communication entre les deux machines au niveau AAL se fait via une liaison virtuelle que l'on peut voir à la *Figure 5-7* ; cette connexion est représentée au milieu de la figure. On peut remarquer que les trois AAL-PDU's sont émis de manière synchrone parce que le type de protocole AAL qui a été choisi est l'AAL de type 1. On peut voir également la seconde connexion qui se trouve dans la partie gauche de la représentation du réseau.

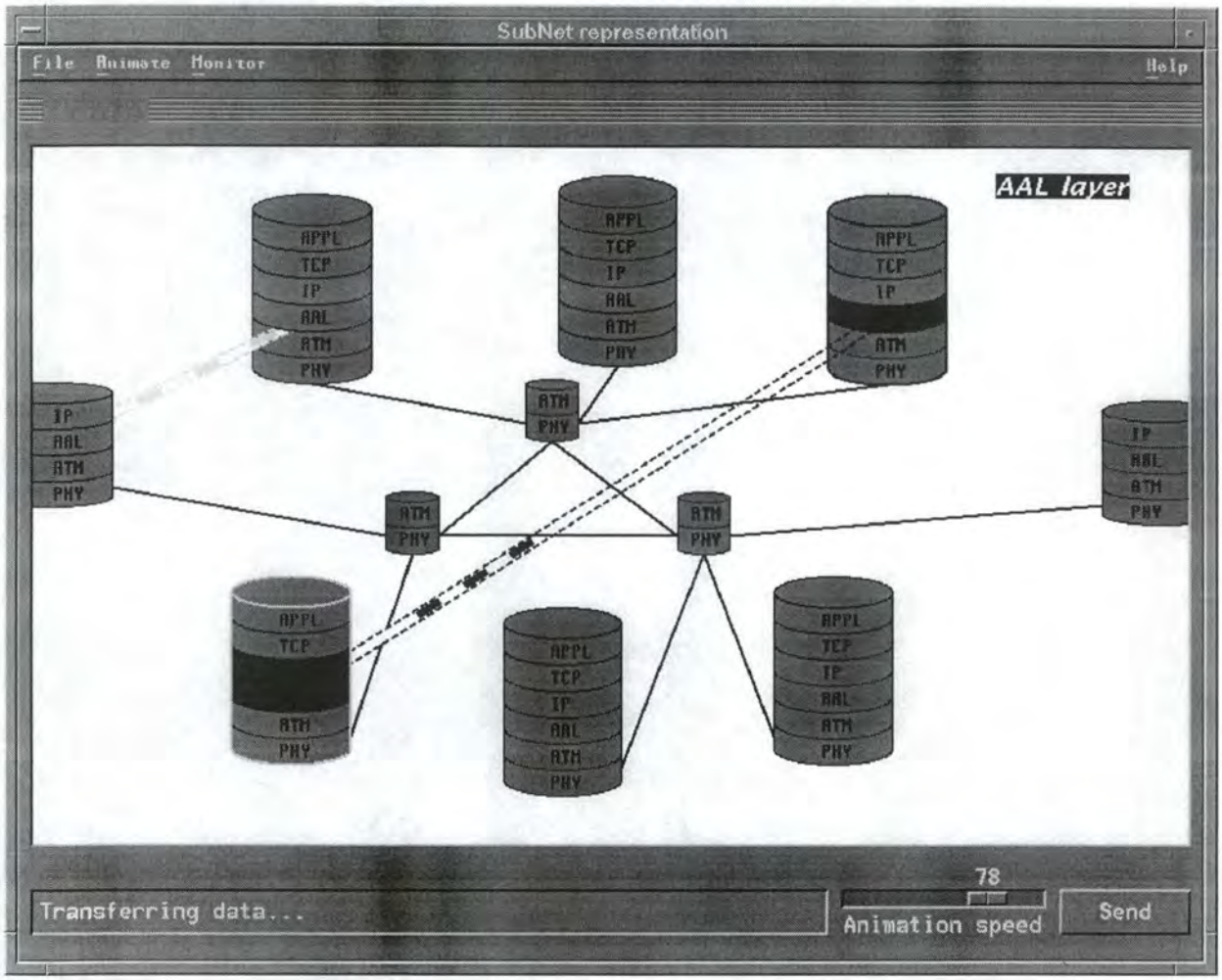


Figure 5-7 : Connexion virtuelle entre les couches AAL des machines sélectionnées

La création des AAL-PDU's se déroule dans la fenêtre dédiée à cet effet et passe par différentes étapes. Ces étapes ont été amplement spécifiées dans le chapitre précédent traitant des spécifications de notre produit. Nous n'allons donc pas illustrer ici toutes les étapes par lesquelles passe le datagramme IP pour être transformé en AAL-PDU's mais seulement deux d'entre elles.

La *Figure 5-8* illustre le passage du datagramme IP à la couche AAL. Celui-ci est alors considéré comme AAL-SDU. La sous-couche CS reçoit le datagramme et lui ajoute les CS header et trailer avant de passer le CS-PDU formé à la sous-couche SAR (Cfr. *Figure 5-9*).

Le CS-PDU passé à la couche SAR est segmenté en morceaux d'une longueur déterminée par le type de protocole AAL choisi par l'utilisateur. A ces morceaux sont ensuite ajoutés des champs spécifiques au type de protocole AAL. Ces champs sont regroupés dans notre représentation sous les termes SAR header et SAR trail. A la fin de ces étapes, comme c'est illustré à la *Figure 5-9*, les AAL-PDU's sont prêts pour être envoyés sur la liaison virtuelle créée.

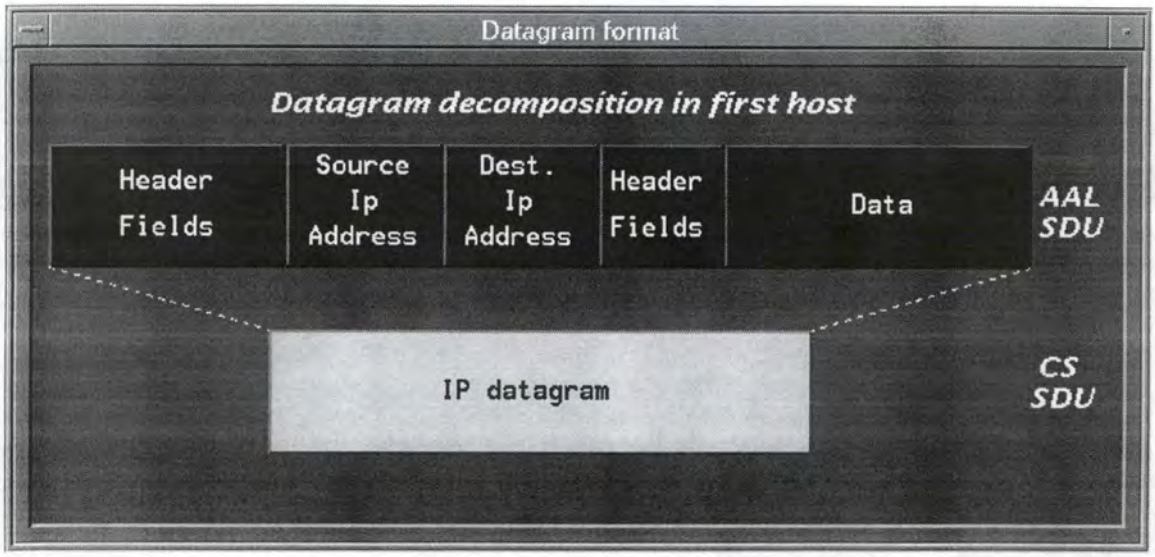


Figure 5-8 : Passage du datagramme IP à la couche AAL

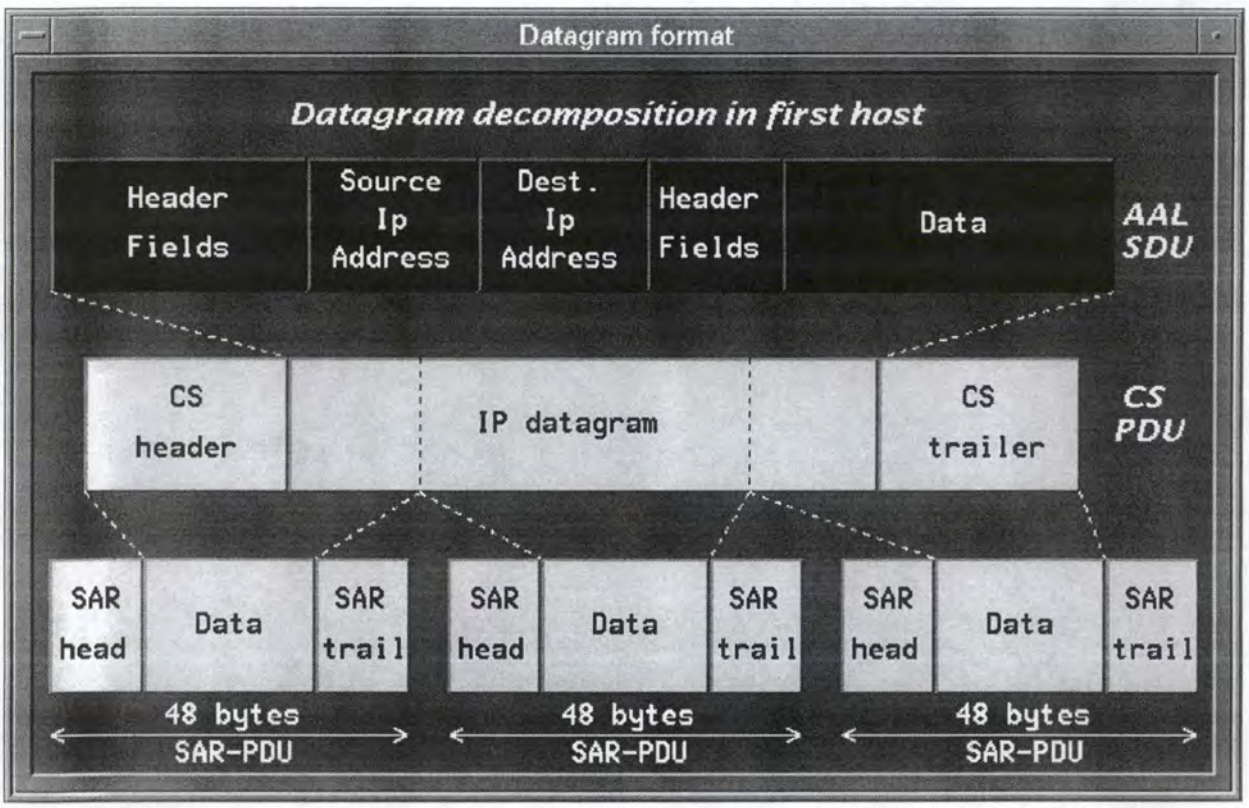


Figure 5-9 : Passage du datagramme IP aux AAL-PDU's

Une fois l'animation lancée, nous avons choisi d'afficher dans cette fenêtre les formats des CS-PDU et SAR-PDU propres au type de protocole AAL sélectionné. La *Figure 5-10* illustre le format choisi pour l'AAL de type 3/4.

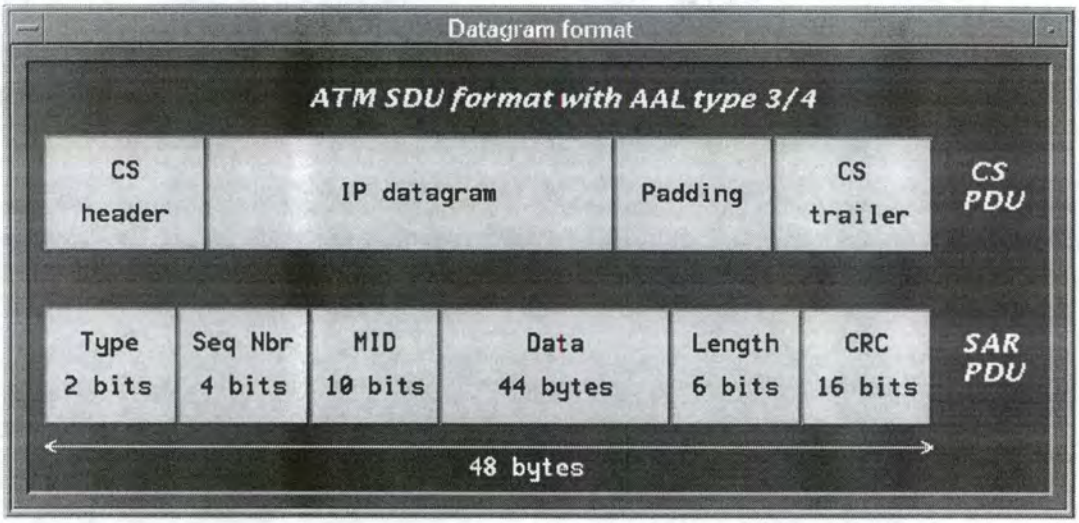


Figure 5-10 :Format des CS-PDU et SAR-PDU pour l'AAL de type 3/4

5.2.3. La couche ATM

Après avoir illustré ce qui se passe lors de l'animation au niveau AAL, nous allons illustrer par quelques figures, certaines fonctions de la couche ATM. La *Figure 5-11* illustre l'animation au niveau ATM. On peut voir les connexions virtuelles créées ainsi que les cellules empruntant ces connexions.

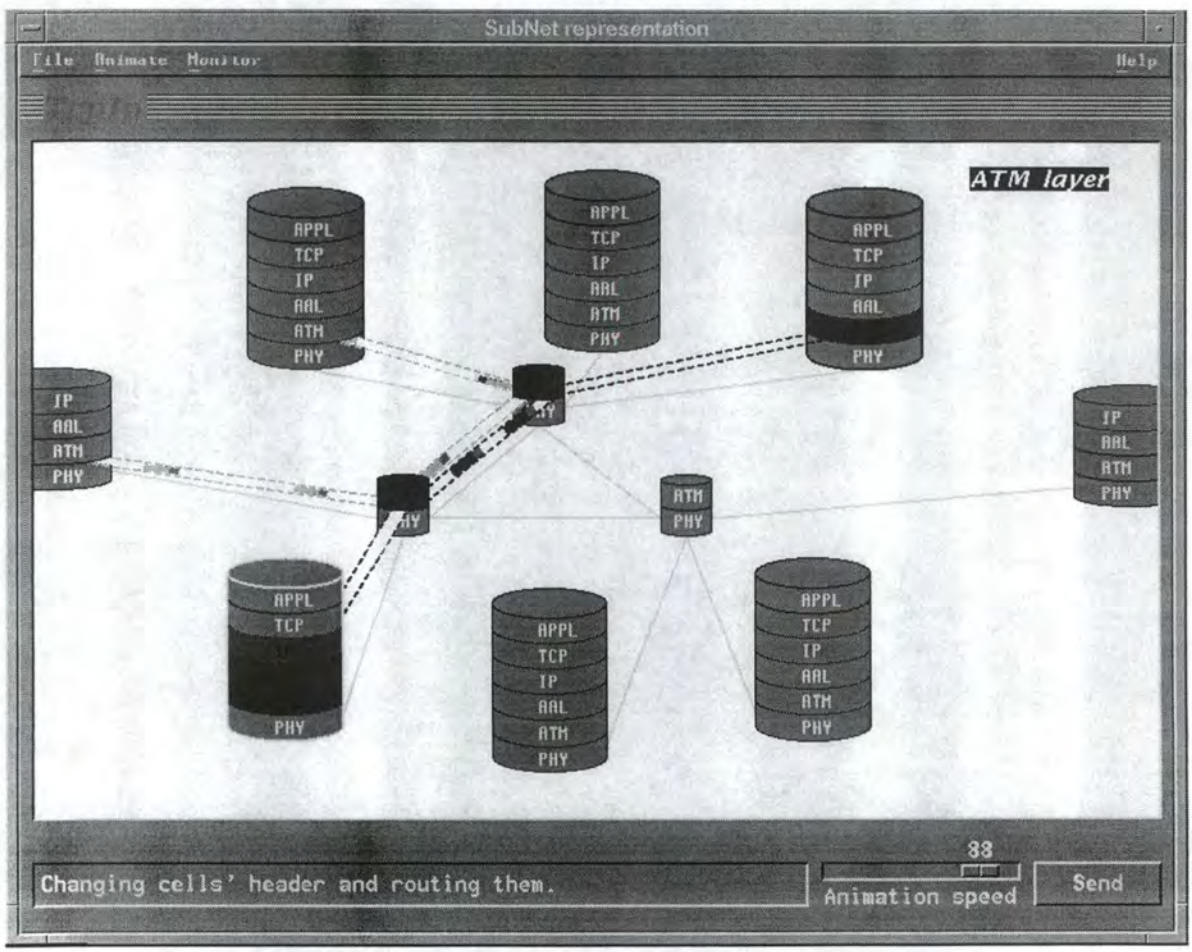


Figure 5-11 : Animation au niveau de la couche ATM

La création des cellules peut être observée par l'utilisateur dans la fenêtre dédiée au format des PDU's. La construction de la cellule passe par plusieurs étapes illustrées dans le didacticiel. Nous n'illustrons ici que deux de ces étapes.

Le passage du AAL-PDU à la couche ATM est illustré à la *Figure 5-12*. Ce AAL-PDU constitue le champ d'informations de la cellule auquel est ajouté un en-tête comme cela est illustré à la *Figure 5-13*. Nous avons voulu montrer les valeurs des champs VPI et VCI dans cet en-tête en mettant en toile de fond ces numéros identifiants tels qu'ils sont repris dans la table de routage. L'élève peut donc voir le même numéro dans la table de routage et dans la représentation de la cellule. Ces numéros changeront lorsque la cellule passe dans chaque noeud du réseau.

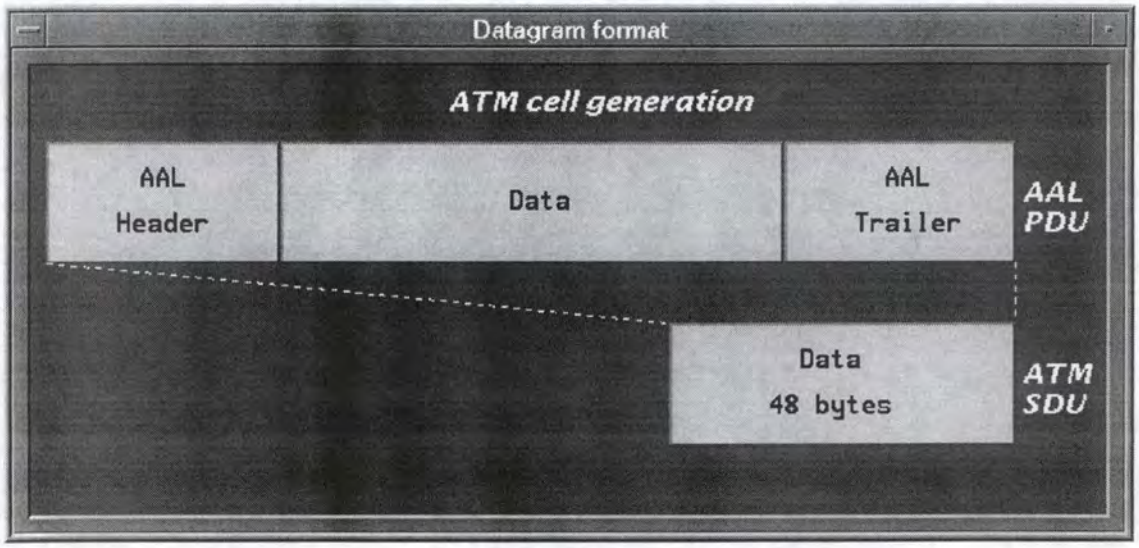


Figure 5-12 : Passage de l'AAL-PDU à la couche ATM

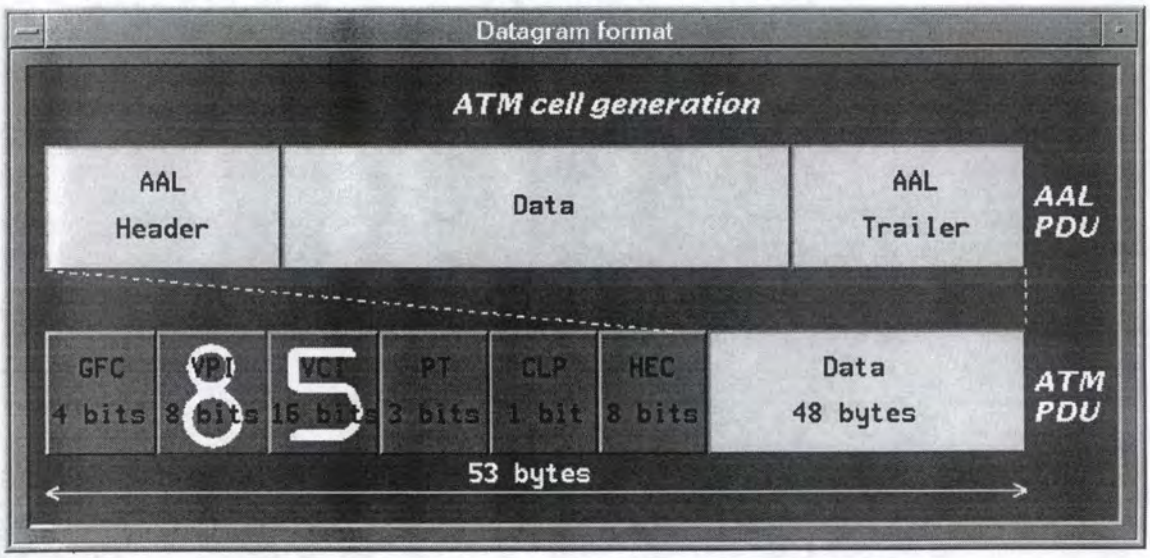


Figure 5-13 : Phase de construction de l'en-tête de la cellule

Il reste à présenter la table de routage qui matérialise une des principales fonctions de la couche ATM. L'interface utilisée pour cette table de routage est assez conforme à la manière dont nous l'avions spécifiée au chapitre précédent. La *Figure 5-14* illustre la phase de création de cette table de routage. Lorsqu'un lien virtuel est établi, comme c'est le cas ici entre les deux noeuds du réseau qui interviennent dans la connexion entre les machines sélectionnées, les VCI et VPI des deux noeuds sont mis en évidence.

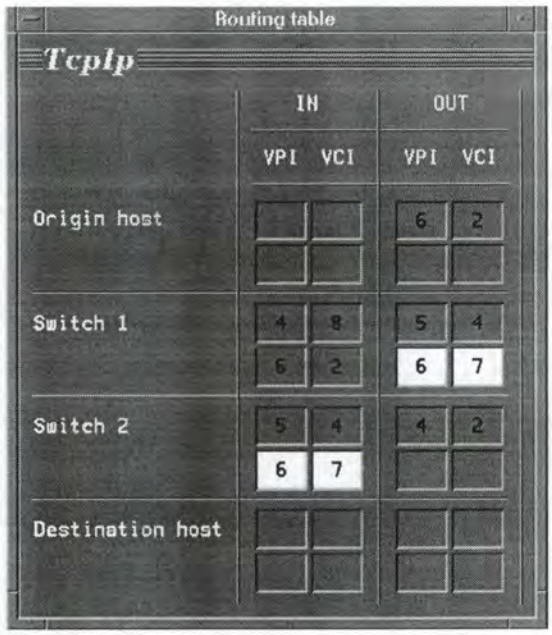


Figure 5-14 : Table de routage (phase de création)

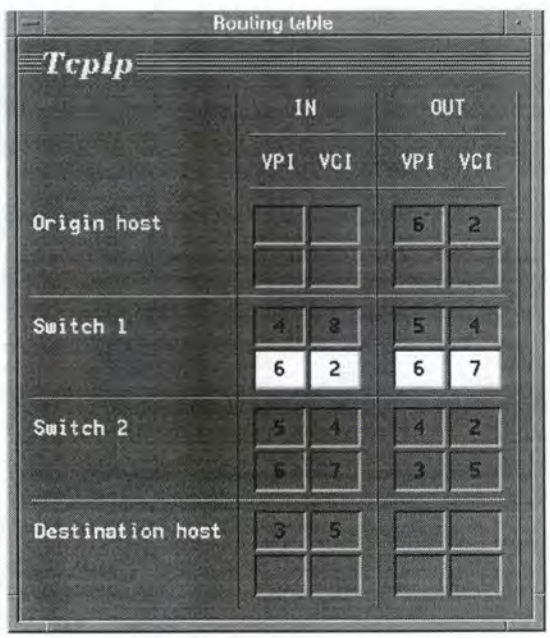


Figure 5-15 : Table de routage (phase de consultation)

Lors de la phase de consultation, ce sont les couples VPI/VCI du même noeud qui sont mis en évidence. Le couple VPI/VCI de sortie (*out*) remplace le couple d'entrée (*in*) dans l'en-

tête des cellules. La valeur de ces identifiants se retrouve en toile de fond dans leur champ respectif VPI et VCI sur la représentation graphique du format de la cellule comme on a pu le voir à la *Figure 5-12*.

5.2.4. La couche physique

Après avoir illustré l'animation au niveau des deux couches supérieures du protocole, nous allons illustrer l'animation au niveau physique. La seule figure que nous montrons est celle de l'animation dans la fenêtre du sous-réseau ATM (*Figure 5-16*). Le format des cellules est le même qu'au niveau de la couche ATM. Pour cette raison, la représentation graphique des cellules est identique à celle proposé à la *Figure 5-12* mais sans les numéros identifiants VPI/VCI en toile de fond. Toutes les fonctions de cette couche physique sont illustrées dans la fenêtre principale du sous-réseau.

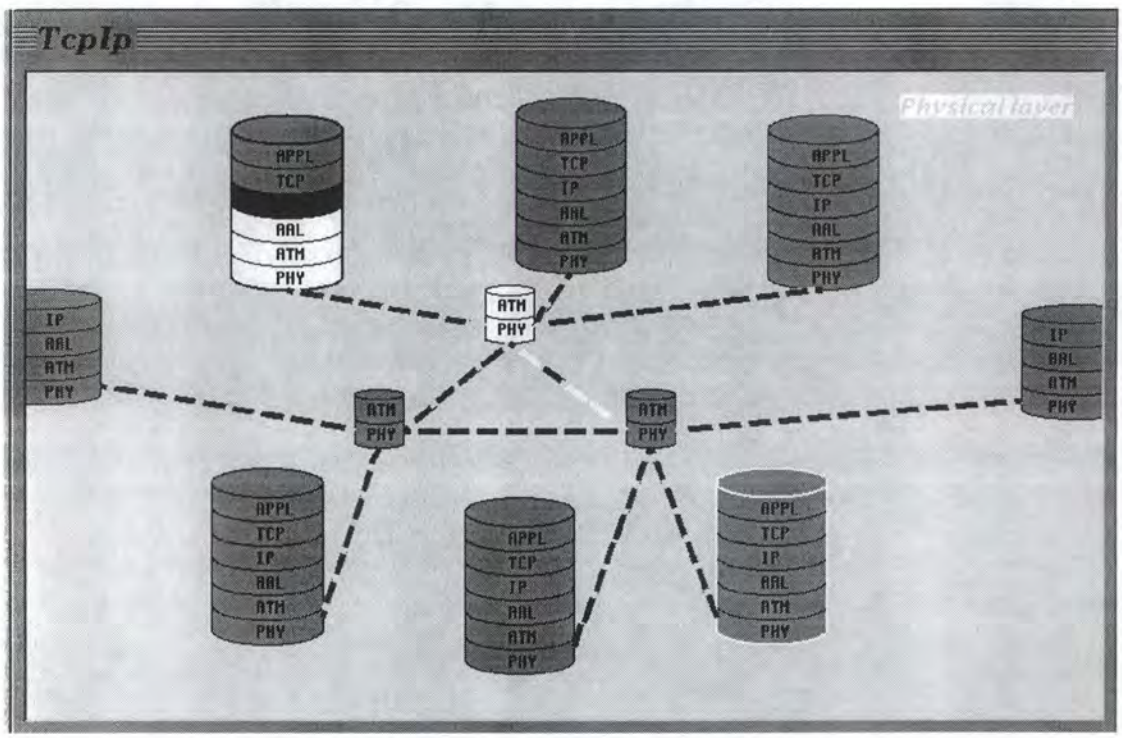


Figure 5-16 : Animation au niveau physique

On remarque sur le dessin de l'animation que la fonction d'adaptation au débit est réalisée entre les deux noeuds du réseau par lesquels passent les deux connexions. Les cellules des deux connexions sont entrelacées sur la liaison virtuelle reliant ces deux noeuds.

5.3. L'architecture logicielle

Après avoir consacré le chapitre précédent aux spécifications de notre logiciel, nous allons fournir dans cette partie une description détaillée de l'architecture logicielle que nous avons adoptée.

La partie logicielle que nous avons développée s'insère dans un logiciel existant. Pour cette raison nos choix en matière de structuration logicielle et de langages de programmation sont limités. Nous avons donc étendu la structure du logiciel en y intégrant de nouveaux modules et de nouveaux objets. La plupart de ces nouveaux objets sont dérivés des objets existant dans TcpIp.

Le langage de programmation utilisé pour la réalisation de cette partie consacrée à la hiérarchie d'objets est écrite en C++. Le reste du code est écrit en C en utilisant les librairies OSF Motif pour réaliser l'interface sous Xwindow. Les ouvrages de référence que nous avons utilisés pour la programmation sont [KER88], [YOU94], [STR95] et [MEY92]. La création des interfaces a été facilitée par l'utilisation de l'outil UI2C. C'est un générateur de code automatique développé par Daniel Muller au CEGELY. Cet outil est spécifié dans [MUL94].

5.3.1. Le graphe d'héritage et relations entre les classes d'objets

Le graphe que nous présentons à la *Figure 5-17* représente le graphe d'héritage des objets qui existent dans TcpIp actuellement. Ce graphe se base sur celui développé par Xavier Gobert dans [XGO95][6] auquel viennent se greffer les nouvelles classes. A ce graphe d'héritage, nous avons ajouté les relations qui existent entre les objets.

La classe centrale de la hiérarchie est évidemment la classe internet qui représente l'Internet. L'Internet est composé de réseaux. Ceux-ci étaient représentés initialement par la classe subnet mais actuellement, ils sont représentés par les classes qui sont des spécialisations de la classe subnet. Cette dernière reste cependant le noeud auquel viennent se lier les classes link et host ou celles qui en sont des spécialisations. L'héritage permet à toutes les classes dérivées de la classe subnet de posséder également ces liens avec les classes link et host ou celles qui en sont dérivées.

La classe link sert de parent à la classe link_ATM qui elle-même est parent de la classe VP. Les deux classes dérivées de la classe link correspondent aux liens ATM et c'est pour cette raison que la classe cell qui représente les cellules ATM est reliée à la classe link_ATM. De cette manière, la classe cell est aussi reliée indirectement à la classe VP qui hérite de la classe link_ATM.

La classe host définie dans TcpIp dans sa version initiale sert de base aux classes définies pour la partie ATM de la même manière que la classe link sert de base pour les classes link_ATM et VP. La hiérarchie des nouvelles classes liées au protocole ATM correspond à une gradation entre les éléments d'un réseau ATM. La classe switch_ATM est directement dérivée de la classe host. Elle représente les noeuds ATM. Ensuite, la classe host_ATM qui en hérite est celle qui représente les machines simples liées au réseau ATM. Et enfin la classe la plus spécialisée est celle nommée gate_ATM qui représente une passerelle ATM.

[6] p. 93

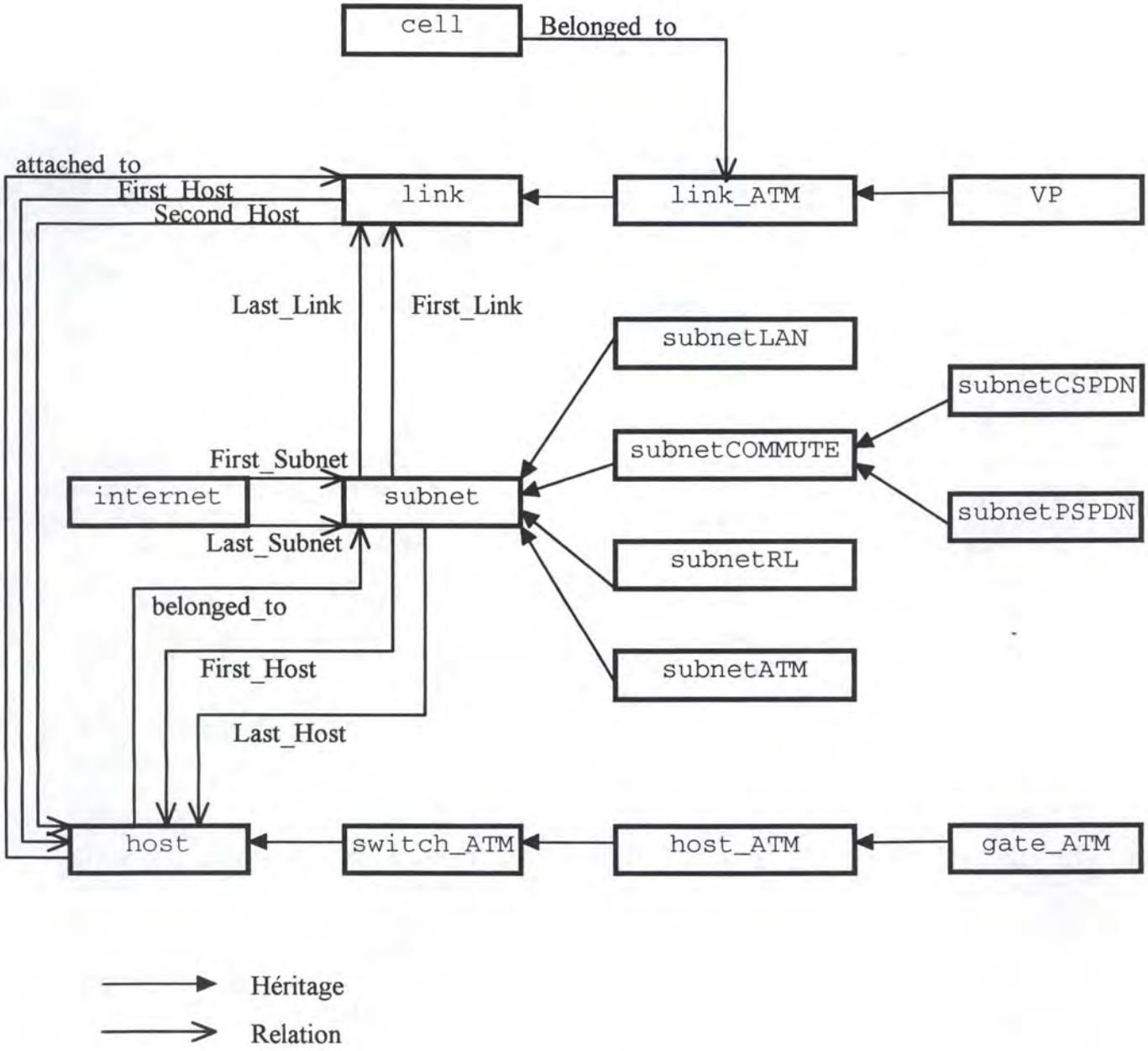


Figure 5-17 : Graphe d'héritage et relations entre les classes d'objet

5.3.2. Les objets

Avant de décrire les objets que nous avons créés spécifiquement pour répondre aux besoins de la partie ATM du logiciel TcpIp, nous allons d'abord décrire les changements que nous avons apportés à la structure existante pour profiter davantage des possibilités de la philosophie objet.

Les objets initiaux créés dans TcpIp décrivent un Internet (classe internet), un sous-réseau (classe subnet), une machine (classe host) ou un lien (classe link). Nous avons changé la classe subnet de manière à pouvoir être plus conforme à la philosophie objet et pouvoir bénéficier plus facilement des avantages de l'héritage.

classe subnet de manière à pouvoir être plus conforme à la philosophie objet et pouvoir bénéficier plus facilement des avantages de l'héritage.

Chaque fois qu'une méthode a besoin des paramètres d'une fenêtre graphique, nous mettons "paramètres graphiques" dans les arguments. Nous regroupons donc sous ce terme, un *display*, un *drawable* et un *GC* (*graphic context*). Ces trois termes sont propres aux libraires de OSF Motif [YOU94]. Lorsqu'on ajoute à ces paramètres graphiques un paramètre représentant un *Widget*, on parlera de "paramètres graphiques étendus".

Dans toutes les méthodes des objets que nous allons décrire, il ne sera jamais fait mention d'une précondition imposant que l'objet existe. C'est une des caractéristiques de la programmation orientée objet : toutes ces méthodes ne peuvent être appelées qu'à partir d'un objet ce qui implique directement l'existence de l'objet.

Nous allons maintenant décrire brièvement les classes d'objets que nous avons introduites uniquement pour rendre la programmation et l'intégration future plus aisée. Ces classes sont dérivées de la classe subnet. Cette dernière a été fortement modifiée par rapport à ce qu'elle incluait dans la version initiale de TcpIp. Elle est devenue virtuelle pure, c'est-à-dire qu'aucun objet de cette classe ne peut être instancié ; seuls des objets de classes dérivées peuvent être instanciés.

I. La classe "subnet"

La classe subnet est une des classes de base de TcpIp. Cependant, pour pouvoir bénéficier pleinement des possibilités offertes par la programmation orientée-objet, nous lui avons fait subir des modifications. Tout d'abord, cette classe est devenue virtuelle pure en ce sens qu'aucun objet de ce type ne peut être créé. Seules les classes dérivées peuvent être instanciées. Ensuite, un certain nombre d'attributs et de méthodes ont été ajoutés. Et enfin, d'autres méthodes ont été modifiées pour remplir sa nouvelle fonction de classe virtuelle pure.

① *Structure de données*

Les attributs nouveaux qui ont été ajoutés sont les suivants :

- *pixmap, bitmap* : attributs monovalués représentant le motif utilisé pour réaliser le fond du sous-réseau lorsque la vue demandée est celle de l' "*Internet comme un réseau de réseaux*" et lorsque le fond demandé pour la vue des sous-réseaux est non coloré.
- *subnetBG* : attribut monovalué représentant la couleur de fond utilisée dans la vue de l'"*Internet comme un réseau de réseaux*" et lorsque le fond demandé pour la vue des sous-réseaux est coloré.
- *minWidth, minHeight* : attributs monovalués indiquant les dimensions minimales que peut prendre la fenêtre du sous-réseau.
- *realWidth, realHeight* : attributs monovalués indiquant les dimensions réelles de la fenêtre du sous-réseau.
- *CONS* : attribut monovalué indiquant si le sous-réseau est orienté connexion ou pas (*CONS: Connection Oriented Network Service)*.

② *Spécification des méthodes*

Les méthodes que nous allons décrire ci-dessous sont uniquement celles qui ont été ajoutées à celles qui existaient déjà et celles qui ont été modifiées. Pour les spécifications des autres méthodes, se référer à [XGO95].

- La méthode "subnet" (constructeur)

  Arguments : une structure de type subnet_creator_parameter (nombre d'hôtes, nombre de liaisons, adresse IP du réseau, nom standard des machines du réseau, sens de présentation, coordonnées relatives et dimensions relatives).
  Préconditions : ce constructeur ne peut être appelé que d'un constructeur d'une classe dérivée de cette classe virtuelle pure qui ne peut avoir aucun objet subnet instancié.
  Postconditions : les attributs de l'objet correspondant aux arguments sont initialisés aux valeurs de ces arguments, tous les éléments de la table de routage sont initialisés à la valeur 0 et tous les pointeurs sont initialisés à la valeur nulle (NULL).

- La méthode "trace"

  Arguments : les paramètres graphiques étendus.
  Préconditions : aucune.
  Postconditions : un rectangle représentant la surface du sous-réseau dans la vue de l'"*Internet comme un réseau de réseaux*" est tracé avec un effet 3-D. Ce rectangle est dessiné dans la couleur subnetBG ou avec le motif de fond bitmap selon que l'utilisateur souhaite ou pas avoir un fond coloré pour la représentation des réseaux.

- La méthode "traceInSubnet"

  Arguments : les paramètres graphiques étendus.
  Préconditions : aucune.
  Postconditions : aucune puisque c'est une méthode purement générique. Le but de cette méthode est de dessiner la représentation du réseau dans la fenêtre utilisée pour la vue détaillée des sous-réseaux.

- La méthode "trace3D"

  Arguments : paramètres graphiques étendus, couleur du haut, couleur du bas.
  Préconditions : aucune.
  Postconditions : quatre lignes sont tracées autour d'un rectangle pour donner l'impression d'un effet 3-D.

- La méthode "GetDimension"

  Arguments : référence à un tableau de 4 entiers.
  Préconditions : aucune.

Postconditions : le tableau contient les valeurs des 4 attributs minWidth, minHeight, realWidth et realHeight dans cet ordre.

- La méthode "whichP"

Arguments : 2 widgets.
Préconditions : le premier widget est de type XmText et le second est de type XmFrame.
Postconditions : aucune puisque c'est une méthode purement virtuelle.

- La méthode "ChooseDatagramFormat"

Arguments : aucun.
Préconditions : aucune.
Postconditions : aucune puisque c'est une méthode purement virtuelle.

Les méthodes "Compute_Way_in_InternetH",
"Compute_Way_in_InternetV",
"Compute_Way_in_SubnetH",
"Compute_Way_in_SubnetV".

Arguments : aucun.
Préconditions : aucune.
Postconditions : aucune puisque ces méthodes sont purement virtuelles.

Les méthodes ~subnet, recompute, host_search (les deux méthodes), merge_host, GetValue et construct_routing_table n'ont pas été modifiées et sont héritées telles quelles dans les classes dérivées.

II. Les classes "subnetLAN" et "subnetRL"

A partir de la classe virtuelle pure subnet, nous avons défini plusieurs classes d'objets qui sont des spécialisations de celle-ci. Elles présentent toutes la même structure de données et les mêmes méthodes dont le contenu seul change. La seule classe qui soit différente est subnetCOMMUTE qui est elle-même virtuelle pure.

Comme les spécifications des méthodes héritées sont les mêmes pour plusieurs classes, nous n'écrirons ces spécifications que pour les méthodes d'une seule classe. Donc, les méthodes non spécifiées des classes décrites plus loin possèdent la même spécification que la méthode de la classe subnetLAN de même nom prise comme exemple.

Les spécifications des méthodes des classes subnetLAN et subnetRL sont identiques.

- La méthode "subnetLAN" (constructeur)

Arguments : une structure de type subnet_creator_parameter (nombre d'hôtes, nombre de liaisons, adresse IP du réseau, nom standard des machines du réseau, sens de présentation, coordonnées relatives et dimensions relatives).
Préconditions : aucune.

Postconditions : un nouvel objet subnetLAN est créé. Ses attributs correspondant aux arguments sont initialisés par l'appel au constructeur de la classe subnet. Ses attributs bitmap, pixmap, minWidth, minHeight, realWidth, realHeight, subnetBG, CONS, Protocol sont initialisés avec des valeurs spécifiques à la classe subnetLAN. De plus, tous les objets host et link composant le sous-réseau sont également créés et rattachés à l'objet subnetLAN par les listes chaînées représentées par les attributs First_Host, Last_Host, First_Link, Last_Link.

- La méthode "trace"

Arguments : les paramètres graphiques étendus.
Préconditions : aucune.
Postconditions : le rectangle formant les délimitations du sous-réseau dans l'Internet est dessiné en 3-D par l'appel à la méthode trace de la classe subnet. Les représentations des objets host liés à cet objet subnetLAN sont également dessinées par l'appel de la méthode trace de ces objets .

- La méthode "traceInSubnet"

Arguments : les paramètres graphiques étendus.
Préconditions : aucune.
Postconditions : la représentation de l'objet est dessinée dans la fenêtre dédiée à la vue détaillée du sous-réseau. Les représentations des objets host et link sont également dessinées par l'appel de leur méthode trace respective.

- La méthode "whichP"

Arguments : 2 widgets.
Préconditions : le premier widget est de type XmText et le second est de type XmFrame.
Postconditions : le texte associé au widget de type XmText est le nom du réseau représenté par cet objet. Si le fond pour la représentation des réseaux est coloré, la couleur de fond du widget de type XmFrame prend la couleur subnetBG. Sinon, l'attribut pixmap est assigné comme fond au widget.

- La méthode "ChooseDatagramFormat"

Arguments : aucun.
Préconditions : aucune.
Postconditions : le type de datagramme correspondant à la classe à laquelle cet objet appartient est assigné à la variable WhichDatagram qui sert à savoir à tout moment quel est le format du datagramme à afficher dans la fenêtre prévue à cet effet. De plus, le message approprié est affiché dans la zone de commentaire de la fenêtre principale.

- Les méthodes "Compute_Way_in_InternetH",
  "Compute_Way_in_InternetV"

Arguments : aucun.
Préconditions : aucune.

Postconditions : le chemin graphique que les paquets devront emprunter dans le réseau considéré dans une vue de l'Internet est calculé et les coordonnées des segments de ce chemin sont mises dans la variable internet_way.

- `Les méthodes "Compute_Way_in_SubnetH", "Compute_Way_in_SubnetV"`

Arguments : aucun.
Préconditions : aucune.
Postconditions : le chemin graphique que les paquets devront emprunter dans le réseau représenté dans sa vue détaillée est calculé et les coordonnées des segments de ce chemin sont mises dans la variable subnet_way.

III. La classe "subnetCOMMUTE"

Cette classe permet de généraliser des méthodes identiques pour toutes les classes caractérisées par le principe de commutativité (subnetCSPDN et subnetPSPDN). Elle possède un attribut supplémentaire. Nous allons uniquement spécifier les méthodes dont les postconditions changent. Les autres sont conformes aux spécifications faites pour la classe subnetLAN ou la classe subnet.

① *Structure de données*

- *LineStyle* : attribut multivalué indiquant, pour les réseaux CONS, le style (continu ou pointillé) de la ligne tracée qui symbolisera la connexion réelle (dans les CSPDN) ou virtuelle (dans les PSPDN) entre les deux machines.

② *Spécification des méthodes*

- `la méthode "subnetCOMMUTE" (constructeur)`

Arguments : une structure de type subnet_creator_parameter (nombre d'hôtes, nombre de liaisons, adresse IP du réseau, nom standard des machines du réseau, sens de présentation, coordonnées relatives et dimensions relatives).
Préconditions : ce constructeur ne peut être appelé que d'un constructeur d'une classe dérivée ; cette classe étant virtuelle pure, aucun objet subnetCOMMUTE ne peut être instancié.
Postconditions : les seuls attributs minWidth, minHeight, realWidth, realHeight et CONS sont initialisés avec des valeurs spécifiques à la classe subnetCOMMUTE. Les objets host et link composant le sous-réseau sont également créés et rattachés à l'objet subnetCOMMUTE par les listes chaînées représentées par les paramètres First_Host, Last_Host, First_Link, Last_Link.

- `Les méthodes "whichP" et "ChooseDatagramFormat"` de cette classe sont virtuelles pures et n'ont donc aucune postcondition.

IV. Les classes "subnetCSPDN" et "subnetPSPDN"

Les deux classes subnetCSPDN et subnetPSPDN sont dérivées de la classe subnetCOMMUTE. Elles sont parfaitement similaires. Les trois seules méthodes de ces classes qui sont redéfinies sont les deux méthodes virtuelles pures (whichP et ChooseDatagramFormat) de la classe subnetCOMMUTE et le constructeur. Ce constructeur initialise les attributs de la classe dérivée de manière complémentaire avec le constructeur de la classe subnetCOMMUTE. Nous allons spécifier uniquement le constructeur d'une des deux classes (subnetCSPDN) et pour les deux autres méthodes, on s'en référencera aux spécifications fournies pour la classe subnetLAN.

Spécification des méthodes

- La méthode "subnetCSPDN"

Arguments : une structure de type subnet_creator_parameter (nombre d'hôtes, nombre de liaisons, adresse IP du réseau, nom standard des machines du réseau, sens de présentation, coordonnées relatives et dimensions relatives).
Préconditions : aucune.
Postconditions : un nouvel objet subnetCSPDN est créé. Ses attributs correspondant aux arguments sont initialisés par l'appel au constructeur de la classe subnetCOMMUTE. Ses attributs bitmap, pixmap, subnetBG et Protocol sont initialisés avec des valeurs spécifiques à la classe subnetCSPDN. Les objets host et link composant le sous-réseau sont créés lors de l'appel du constructeur de subnetCOMMUTE.

V. La classe "subnetATM"

La classe subnetATM est la nouvelle classe que nous avons ajoutée pour créer le nouveau sous-réseau dans TcpIp. Le principe d'héritage permet de ne définir qu'un nouvel attribut pour cette classe et de bénéficier de toutes les méthodes déjà définies pour les classes génériques. Les méthodes ont été redéfinies comme pour les autres classes dérivées de la classe subnet et leur spécification est identique aux méthodes spécifiées pour la classe subnetLAN. Nous allons uniquement spécifier les méthodes présentant une différence par rapport à la spécification donnée pour la classe subnetLAN.

① *Structure de données*

- *Switch_Nbr* : attribut monovalué indiquant le nombre de noeuds (*switch*) que la représentation du réseau ATM possède.

② *Spécification des méthodes*

- La méthode "subnetATM" (constructeur)

Arguments : une structure de type subnet_creator_parameter (nombre d'hôtes, nombre de liaisons, adresse IP du réseau, nom standard des machines du réseau,

sens de présentation, coordonnées relatives et dimensions relatives), nombre de noeud.
Préconditions : aucune.
Postconditions : un objet de type subnetATM est créé. Tous ses attributs sont initialisés avec les valeurs présentes en argument ou avec des valeurs propres à la classe subnetATM. Toutes les variables globales utilisées pour l'animation dans le réseau ATM sont initialisées. Les objets host_ATM, gate_ATM, switch_ATM et link_ATM sont créés et reliés à cet objet par les listes chaînées représentées par les paramètres First_Host, Last_Host, First_Link, Last_Link.

- La méthode "~subnetATM" (destructeur)

Arguments : aucun.
Préconditions : aucune.
Postconditions : l'objet de type subnetATM est détruit ainsi que tous les objets qui lui sont attachés.

- La méthode "trace"

Arguments : les paramètres graphiques étendus.
Préconditions : aucune.
Postconditions : le rectangle formant les délimitations du sous-réseau dans l'Internet est dessiné en 3-D par l'appel à la méthode trace de la classe subnet. Les représentations des objets host_ATM et gate_ATM liés à cet objet de type subnetATM sont également dessinées par l'appel de leur méthode trace. Par contre, on ne représente pas les objets de type switch_ATM.

- La méthode "recompute"

Arguments : aucun.
Préconditions : aucune.
Postconditions : si la vue n'est pas la vue détaillée du sous-réseau, alors appel de la méthode recompute de la classe subnet.
Sinon, tous les éléments graphiques intervenant dans l'animation sont recalculés en plus des objets liés à cet objet de type subnetATM. Il s'agit des cellules, des liaisons virtuelles et des connexions virtuelles selon le niveau auquel se passe l'animation.

- La méthode "traceInSubnet"

Arguments : les paramètres graphiques étendus.
Préconditions : aucune.
Postconditions : la fenêtre graphique correspondant aux paramètres passés en argument est re-dessinée. Lors de l'animation, pour éviter de re-dessiner tous les éléments de la fenêtre à chaque déplacement de cellules, nous avons utilisé une variable globale (RedrawALL) qui permet de ne re-dessiner que les cellules lorsque c'est suffisant. Si RedrawALL est vrai, on re-dessine tout

en tenant compte du niveau de l'animation qui se déroule; sinon, on ne re-dessine que les cellules.

VI. La classe "cell"

La classe cell est utilisée pour représenter une cellule. On regroupe ici sous le terme cellule, les cellules de la couche ATM, celles de la couche physique mais aussi les PDU's de la couche AAL. Cet objet est représenté à l'écran par un segment de droite épais et coloré.

① *Structure de données*

- *belonged_to* : attribut monovalué pointant vers un objet de type link_ATM qui représente la liaison virtuelle sur laquelle la cellule va circuler. Cet attribut est exclusif avec le suivant. Ou bien une cellule appartient à une liaison virtuelle (c'est le cas pour les AAL-PDU's), ou bien elle appartient à une connexion virtuelle (c'est le cas pour les cellules ATM et physiques).
- *path* : attribut monovalué pointant vers une liste d'objets de type link_ATM. Cette liste correspond à une connexion virtuelle dont chaque élément est une liaison virtuelle.
- *color* : attribut monovalué représentant la couleur avec laquelle la cellule sera dessinée.
- *ATMway* : attribut multivalué représentant la suite des coordonnées de la cellule utilisées pour parcourir entièrement la liaison ou la connexion à laquelle elle appartient. Cet ensemble de coordonnées est de la forme $(x_1,y_1),(x_2,y_2)$.
- *stop* : attribut multivalué représentant les positions dans ATMway qui correspondent à l'arrivée de la cellule dans un noeud du réseau ou à destination.

② *Spécification des méthodes*

- `La méthode "cell" (le constructeur)`

  Arguments : pointeur vers une liaison, pointeur vers une connexion (Un des deux a toujours la valeur "NULL").
  Préconditions : aucune.
  Postconditions : un nouvel objet de type cell est créé. Les deux attributs belonged_to et path sont initialisés avec la valeur des deux paramètres et les autres attributs sont initialisés à 0.

- `La méthode "~cell" (le destructeur)`

  Arguments : aucun.
  Préconditions : aucune.
  Postconditions : l'objet n'existe plus et la zone mémoire allouée pour l'attribut ATMway est libérée.

- `La méthode "TraceInSubnet"`

  Arguments : paramètres graphiques, position (pos) dans le chemin graphique à parcourir.
  Préconditions : aucune.
  Postconditions : si pos $\neq 0$, la représentation de l'objet dessinée aux coordonnées correspondant à la position (pos-1) dans l'attribut ATMway est effacée. La

représentation de l'objet est ensuite dessinée avec la couleur reprise dans l'attribut color, aux coordonnées correspondant à la position pos dans l'attribut ATMway.

- La méthode "TraceInSubnet"[9]

  Arguments : paramètres graphiques, position (pos) dans le chemin graphique à parcourir, couleur de l'en-tête de la cellule (h_color).
  Préconditions : aucune.
  Postconditions : si pos ≠ 0, la représentation de l'objet dessinée aux coordonnées correspondant à la position (pos-1) dans l'attribut ATMway est effacée. La représentation de l'objet est alors dessinée en deux couleurs : la couleur reprise dans l'attribut color pour la représentation du champ d'informations et la couleur h_color pour la représentation de l'en-tête. L'ensemble est dessiné aux coordonnées correspondant à la position pos dans l'attribut ATMway.

- La méthode "PrepareToTrace"

  Arguments : paramètres graphiques, booléen (before) pour indiquer si la fonction est appelée avant TraceInSubnet ou après, largeur (link_width) en pixel de la cellule à dessiner.
  Préconditions : aucune.
  Postconditions : si before est vrai, les lignes qui seront dessinées dans un contexte graphique correspondant à ces paramètres graphiques auront la largeur link_width.
  Si before est faux, les attributs décrivant la manière dont seront dessinées les lignes dans ce contexte graphique sont remis à leur valeur initiale.

- La méthode "ComputeApart"

  Arguments : limite inférieure (inf_limit), limite supérieure (sup_limit), nombre d'itération (nb_iteration), pointeur vers un objet Link_ATM (link). Limite inférieure et limite supérieure sont deux indices déterminant la suite des coordonnées dans ATMway qui vont être calculées par cette méthode. Link_ATM peut être égal à belonged_to ou à une liaison de path.
  Préconditions : nombre d'itération = limite supérieure - limite inférieure.
  Postconditions : les coordonnées successives que la cellule va devoir prendre pour parcourir link sont calculées et placées entre les positions inf_limit et sup_limit dans ATMway.

- La méthode "ComputeATMway_in_AAL"

  Arguments : aucun.
  Préconditions : aucune.
  Postconditions : ATMway contient toutes les coordonnées nécessaires pour dessiner le cheminement de la cellule entre deux couches AAL de deux machines.

[9] Les deux méthodes de même nom sont différenciées en C++ en fonction de leurs arguments

Celles-ci sont reliées par une liaison virtuelle représentée par l'objet sur lequel pointe belonged_to.

- La méthode "ComputeATMway_in_ATM"

Arguments : aucun.
Préconditions : aucune.
Postconditions : ATMway contient toutes les coordonnées nécessaires pour dessiner le cheminement de la cellule entre deux couches ATM de deux machines. Celles-ci sont reliées par une connexion virtuelle représentée par l'objet sur lequel pointe path.

- La méthode "ComputeATMway_in_PHY"

Arguments : aucun.
Préconditions : aucune.
Postconditions : ATMway contient toutes les coordonnées nécessaires pour dessiner le cheminement de la cellule entre deux couches physiques de deux machines. Celles-ci sont reliées par une connexion virtuelle représentée par l'objet sur lequel pointe path.

VII. La classe "switch_ATM"

La classe switch_ATM vise à représenter un noeud du réseau ATM. Cette classe hérite de la classe host existant dans TcpIp. En même temps, elle sert de parent à la classe host_ATM qui elle-même sert de parent à la classe gate_ATM.

① *Structure de données*

- *ATM_ad* : attribut multivalué représentant l'adresse de l'élément du réseau représenté par cet objet au sein du réseau ATM. Cet attribut ne peut avoir deux valeurs que si l'objet auquel il appartient est une passerelle qui relie deux réseaux ATM. Pour les noeuds du réseau et les machines normales, l'adresse ATM est unique.
- *type* : attribut monovalué représentant le type réel de l'objet. Comme l'objet switch_ATM sert de parent à d'autres objets, nous avons utilisé un attribut pour identifier le type réel (SWITCH_ATM, HOST_ATM ou GATE_ATM) de l'objet qui sera référencé par un pointeur sur l'objet générique.
- *linked_to* : attribut multivalué représentant l'ensemble des liens partant de la représentation graphique de l'objet.
- *table* : attribut multivalué représentant l'ensemble des objets switch_ATM, host_ATM et gate_ATM directement accessibles à partir de cet objet.

② *Spécification des méthodes*

- la méthode "switch_ATM" (constructeur)

Arguments : adresse IP, adresse du réseau, nom à attribuer à l'objet, type (HOST ou GATEWAY), position relative de la machine dans le sous-réseau, connexion, adresse ATM, type de machine ATM (attribut type).
Préconditions : aucune.
Postconditions : un objet de type switch_ATM est créé. Ses attributs hérités sont initialisés avec les valeurs des premiers paramètres du constructeur et ses attributs propres (ATM_ad et type) sont initialisés avec les deux derniers paramètres. Les deux attributs table et linked_to sont initialisés à une valeur nulle.

`La méthode "~switch_ATM" (destructeur)`

Arguments : aucun.
Préconditions : aucune.
Postconditions : l'objet de type switch_ATM est détruit et la mémoire utilisée est libérée.

`La méthode "TraceInSubnet"`

Arguments : les paramètres graphiques étendus.
Préconditions : aucune.
Postconditions : la représentation de l'objet est dessinée sur la fenêtre graphique dont les caractéristiques sont passées en paramètre. La position et les dimensions de la représentation sont données par ses attributs x_real, y_real, h_width et h_height.
Le fond de la représentation est dessiné avec la couleur ATMHostColor. Les contours sont dessinés dans la couleur neutre ATMneutral si l'animation se déroule au niveau AAL et dans la couleur ATMfg sinon. Les couches ATM et physique sont aussi mises en évidence avec la couleur ATMcolor pendant certaines étapes de l'animation. Pour décrire les étapes entre lesquelles les couches sont mises en évidence, nous nous référençons à la décomposition en étapes établie au chapitre précédent.

Si l'objet correspond au **premier** noeud dans la connexion entre la machine émettrice et la machine de destination (*Tableau 5-1*).

Tableau 5-1 : Etapes de mise en évidence des couches dans la représentation de l'objet switch qui correspond au premier noeud dans la connexion

ANIMATION		MISE EN EVIDENCE DE	MISE EN EVIDENCE A
Animation au niveau ATM :	couche ATM	5	16.2
Animation au niveau Physique :	couche ATM	12	26
	couche PHY	10	26

Si l'objet correspond au **second** noeud dans la connexion entre la machine émettrice et la machine de destination (*Tableau 5-2*).

Tableau 5-2 : Etapes de mise en évidence des couches dans la représentation de l'objet switch qui correspond au second noeud dans la connexion

ANIMATION		MISE EN EVIDENCE DE	MISE EN EVIDENCE A
Animation au niveau ATM :	couche ATM	7.1	16.2
Animation au niveau Physique :	couche ATM	18.3	26
	couche PHY	18.1	26

- La méthode "recompute"

Arguments : la position absolue de la représentation du sous-réseau auquel l'objet est relié, les dimensions réelles de la fenêtre graphique dans laquelle l'objet est représenté, l'adresse IP du sous-réseau, l'adresse de l'objet subnet auquel cet objet est relié.
Préconditions : aucune.
Postconditions : les dimensions réelles h_width et h_height de la représentation de l'objet sont calculées ainsi que ses coordonnées réelles (x_real et y_real). L'adresse IP et l'adresse de l'objet représentant le sous-réseau ne servent à rien ici mais nous les avons gardées pour permettre le polymorphisme de cette méthode. Pour redéfinir une méthode d'un objet parent, il faut que le nombre de paramètres et que les types de ces paramètres soient les mêmes.

- La méthode "computeRealValues"

Arguments : les dimensions absolues de la représentation de l'objet.
Préconditions : aucune.
Postconditions : les dimensions réelles h_width et h_height de la représentation de l'objet sont calculées.

VIII. La classe "host_ATM"

La classe host_ATM représente un host classique dans le réseau ATM. Cette classe hérite de la classe switch_ATM. Les méthodes TraceInSubnet et recompute sont redéfinies et une nouvelle méthode traceCommonPart vient s'ajouter aux deux méthodes précédentes et aux constructeur et destructeur.

② *Spécification des méthodes*

- `la méthode "host_ATM" (constructeur)`

  Arguments : adresse IP, adresse du réseau, nom à attribuer à l'objet, type utilisé dans TcpIp (HOST ou GATEWAY), position relative de la machine dans le sous-réseau, connexion, adresse ATM, type de machine ATM (attribut type).
  Préconditions : aucune.
  Postconditions : un objet host_ATM est créé.

- `la méthode "~host_ATM" (destructeur)`

  Arguments : aucun.
  Préconditions : aucune.
  Postconditions : l'objet de type host_ATM est détruit.

- `La méthode "traceCommonPart"`

  Arguments : les paramètres graphiques étendus, un entier (plus) servant dans le calcul de l'ordonnée du nom des couches dans la représentation des machines et des passerelles, un réel (divide) servant dans le calcul de l'abscisse du nom de ces mêmes couches. Ces deux derniers paramètres sont différents si l'objet qui appelle cette méthode est du type host_ATM ou gate_ATM (voir ci-dessous).
  Préconditions : divide $\neq 0$.
  Postconditions : le fond de la machine est tracé avec la couleur ATMHostColor. Le contour est dessiné dans la couleur appropriée (selected1 si la machine est l'origine sélectionnée par l'utilisateur, selected2 si la machine est la destination sélectionnée par l'utilisateur, blinkingColor si la machine est la passerelle suivante sur le chemin dans l'Internet menant de la source à la destination, ATMfg sinon). Les noms des couches sont écrits dans chacune d'elles aux coordonnées réelles calculées à partir des dimensions réelles de la machine et des deux paramètres plus et divide. Finalement, certaines couches sont mises en évidence. Cette mise en évidence se fait en utilisant la couleur datagramColor pour la couche IP et la couleur ATMcolor pour les autres couches. Elle correspond à la période durant laquelle la couche intervient dans la communication au niveau choisi par l'utilisateur. Cette période est établie pour chaque couche et chaque niveau d'animation en fonction de la décomposition en étapes, établie au chapitre 4.

Dans la machine émettrice (*Tableau 5-3*) :

Tableau 5-3 : Mise en évidence des couches de la machine origine d'après les scénarios de décomposition de la communication en étapes

ANIMATION		MISE EN EVIDENCE DE	MISE EN EVIDENCE A
Animation au niveau AAL :	couche IP	1	11
	couche AAL	2	11
Animation au niveau ATM :	couche IP	1	16.2
	couche AAL	2	16.2
	couche ATM	4	16.2
Animation au niveau Physique :	couche IP	1	26
	couche AAL	2	26
	couche ATM	4	26
	couche PHY	6	26

Dans la machine destinatrice (*Tableau 5-4*) :

Tableau 5-4 : Mise en évidence des couches de la machine destination d'après les scénarios de décomposition de la communication en étapes

ANIMATION		MISE EN EVIDENCE DE	MISE EN EVIDENCE A
Animation au niveau AAL :	couche IP	14	15
	couche AAL	4	15
Animation au niveau ATM :	couche IP	16.6	16.7
	couche AAL	16.4	16.7
	couche ATM	8	16.7
Animation au niveau Physique :	couche IP	25	26
	couche AAL	23	26
	couche ATM	21	26
	couche PHY	19	26

- `La méthode "TraceInSubnet"`

Arguments : les paramètres graphiques étendus.
Préconditions : aucune.
Postconditions : idem que pour traceCommonPart qui est appelée avec les paramètres graphiques étendus et les valeurs 7 et 2.5 pour les paramètres plus et divide.

- La méthode "recompute"

Arguments : la position absolue de la représentation du sous-réseau auquel l'objet est relié, les dimensions réelles de la fenêtre graphique dans laquelle l'objet est représenté, l'adresse IP du sous-réseau, l'adresse de l'objet subnet auquel cet objet est relié.
Préconditions : aucune.
Postconditions : les dimensions réelles h_width et h_height de la représentation de l'objet sont calculées ainsi que ses coordonnées réelles (x_real et y_real). L'adresse IP et l'adresse de l'objet représentant le sous-réseau ne servent à rien ici (même remarque que pour la méthode recompute de la classe switch_ATM).

IX. La classe "gate_ATM"

La classe gate_ATM est utilisée pour représenter une passerelle ATM. Ces passerelles sont d'abord des machines ATM, c'est pourquoi cette classe hérite de la classe représentant les machines simples (host_ATM). Les passerelles possèdent un double protocole. Nous avons donc ajouté un attribut pour conserver le nom de ce second protocole. Les méthodes de la classe gate_ATM sont celles héritées plus le constructeur et le destructeur. Parmi les méthodes héritées, seulement deux sont redéfinies : recompute et traceInSubnet.

① *Structure de données*

- *otherProtocol* : attribut monovalué contenant le nom du protocole du réseau auquel est relié la passerelle représentée par cet objet de type gate_ATM.

② *Spécification des méthodes*

- la méthode "gate_ATM" (constructeur)

Arguments : adresse IP, adresse du réseau, nom à attribuer à l'objet, type utilisé dans TcpIp (HOST ou GATEWAY), position relative de la passerelle dans le sous-réseau, connexion, adresse ATM, type de machine ATM (attribut type), nom du deuxième protocole.
Préconditions : aucune.
Postconditions : un objet de type gate_ATM est créé.

- la méthode "~gate_ATM" (destructeur)

Arguments : aucun.
Préconditions : aucune.
Postconditions : l'objet de type gate_ATM est détruit.

- La méthode "TraceInSubnet"

Arguments : les paramètres graphiques étendus.
Préconditions : aucune.

Postconditions : idem que pour traceCommonPart (méthode héritée de host_ATM). C'est d'ailleurs cette méthode qui est appelée dans deux situations différentes avec les paramètres plus et divide ayant des valeurs différentes. Ceux-ci prennent les valeurs 3 et 2 lorsque la représentation de la passerelle est dessinée sur le bord gauche de la fenêtre et les valeurs 3 et 4,1 lorsque cette représentation est dessinée sur le bord droit. La fenêtre est celle dont les paramètres sont passés en argument.
Rem : les valeurs des paramètres plus et divide ont été obtenues par essais successifs. C'est notamment dans ce genre de situation que certains outils d'aide à la conception d'interfaces auraient été utiles (Cfr. Chapitre 6).

- La méthode "recompute"

Arguments : la position absolue de la représentation du sous-réseau auquel l'objet est relié, les dimensions réelles de la fenêtre graphique dans laquelle l'objet est représenté, l'adresse IP du sous-réseau, l'adresse de l'objet subnet auquel cet objet est relié.
Préconditions : aucune.
Postconditions : les dimensions réelles h_width et h_height de la représentation de l'objet sont calculées ainsi que ses coordonnées réelles (x_real et y_real).
L'adresse IP et l'adresse de l'objet représentant le sous-réseau ne servent à rien ici (même remarque que pour la méthode recompute de la classe switch_ATM).

X. La classe "link_ATM"

La classe link_ATM sert à modéliser un lien entre deux éléments d'un réseau ATM. Cet objet hérite des attributs et méthodes de la classe link. De plus certains attributs ont été ajoutés pour tenir compte des spécificités d'une lien ATM. Une méthode a également été ajoutée, outre les constructeur et destructeur et deux méthodes ont été redéfinies.

① *Structure de données*

- *origin* : attribut monovalué qui représente le point cardinal par lequel la représentation de cet objet est attachée à la représentation de l'objet de type host référencé par l'attribut First_Host.
- *destination* : attribut monovalué qui représente le point cardinal par lequel la représentation de cet objet est attachée à la représentation de l'objet de type host référencé par l'attribut Second_Host.
- *norm* : attribut monovalué représentant la norme du segment de droite formé par les coordonnées réelles de la représentation de l'objet.
- *nbCell* : attribut monovalué ayant pour valeur le nombre de cellules qui pourraient être représentées de manière juxtaposée sur la longueur de ce lien. Cette longueur est contenue dans l'attribut norm et la longueur des cellules est une des constantes de la partie ATM : cellLength

② *Spécification des méthodes*

- la méthode "link_ATM" (constructeur)

  Arguments : premier objet de type host connecté, second objet de type host connecté, coordonnées absolues de la première extrémité, coordonnées absolues de la seconde extrémité, adresse IP de la liaison, orientation de la connexion sur le premier hôte, orientation de la connexion sur le deuxième hôte.
  Préconditions : aucune.
  Postconditions : un nouvel objet de type link_ATM est créé et les attributs de l'objet correspondant aux arguments du constructeur sont initialisés avec la valeur de ceux-ci.

- la méthode "~link_ATM" (destructeur)

  Arguments : aucun.
  Préconditions : aucune.
  Postconditions : l'objet dont le destructeur est appelé est détruit.

- La méthode "traceInSubnet"

  Arguments : les paramètres graphiques.
  Préconditions : aucune
  Postconditions : une ligne représentant un lien physique dans le réseau ATM est tracée avec une largeur de 2 pixels. Ses coordonnées sont x1_real, y1_real, x2_real, y2_real.

- La méthode "recompute"

  Arguments : un protocole et 5 entiers totalement fictifs pour respecter le nombre et les types des arguments de la méthode recompute de la classe link dont elle hérite pour permettre le polymorphisme.
  Préconditions : aucune.
  Postconditions : les coordonnées réelles de la représentation de l'objet sont calculées.

- La méthode "computeNbCell"

  Arguments : aucun.
  Préconditions : cellLength $\neq 0$.
  Postconditions : les attributs norm et nbCell sont calculés.

XI. La classe "VP "

La classe VP devait servir à représenter un Virtual Path au sens ATM mais sa signification a été élargie pour représenter actuellement toute liaison virtuelle au niveau AAL et au niveau ATM. Une liaison virtuelle est représentée par deux droites parallèles dessinées en pointillés pour symboliser le caractère virtuel de cette liaison.

① *Structure de données*

- *d* : attribut monovalué déterminant l'écartement entre les deux lignes composant la représentation d'une liaison virtuelle. Cette valeur est un nombre entier de pixel compris entre 6 et 12. Il est calculé selon la pente de la droite formée par les deux extrémités de cette liaison virtuelle pour garder une perspective 3-D. Si la droite est pratiquement verticale, l'écartement sera maximal et si la droite tend vers l'horizontale, l'écartement sera minimal.
- *xd, yd* : attributs monovalués utilisés pour déterminer les coordonnées réelles des deux droites formant la représentation d'une liaison virtuelle. Les valeurs des attributs x1_real, x2_real, y1_real et y2_real sont les coordonnées utilisées pour dessiner une seule droite. xd et yd permettent de dessiner les deux droites en respectant l'écartement d obtenu.

② *Spécification des méthodes*

- `la méthode "VP" (constructeur)`

  Arguments : les mêmes que ceux du constructeur de link_ATM.
  Préconditions : aucune.
  Postconditions : un nouvel objet du type VP est créé et les attributs de l'objet correspondant aux arguments du constructeur sont initialisés avec la valeur de ceux-ci.

- `la méthode "~VP" (destructeur)`

  Arguments : aucun.
  Préconditions : aucune.
  Postconditions : l'objet dont le destructeur est appelé est détruit.

- `La méthode "traceInSubnet"`

  Arguments : les paramètres graphiques.
  Préconditions : aucune
  Postconditions : deux lignes représentant une liaison virtuelle sont tracées en pointillés. Ces deux lignes sont espacées d'un nombre de pixel correspondant à la valeur de l'attribut d.

- `La méthode "recompute"`

  Arguments : un protocole et 5 entiers pour respecter le nombre et les types des arguments de la méthode recompute de la classe link dont elle hérite pour permettre le polymorphisme. Nous avons utilisé un de ces arguments fictifs comme un booléen. Il s'agit du premier entier.
  Préconditions : aucune.
  Postconditions : les coordonnées réelles de la représentation de l'objet sont calculées. Les attributs d, xd et yd sont également calculés sur base des coordonnées réelles qui viennent d'être calculées.

5.3.3. La découpe en modules

La découpe en modules représente le coeur même de l'architecture logicielle. La découpe utilisée dans TcpIp est basée sur une architecture logique dans laquelle le module constitue une unité de travail pour le concepteur. Nous avons ajouté les modules que nous avons créés à la découpe existante.

Les modules sont organisés en niveaux. Le niveau 5 correspond aux modules de coordination et de traitement. Le niveau 4 regroupe les modules dont les services sont des fonctions d'entrées-sorties. Le niveau 3 est constitué des modules de données [DUB94].

I. La découpe en modules dans TcpIp

Nous avons représenté à la *Figure 5-18* la découpe en modules que l'on a dégagée pour le didacticiel TcpIp. Les modules représentés en pointillés sont ceux qui existaient déjà dans TcpIp[8]. Les autres sont ceux qui ont été ajoutés.

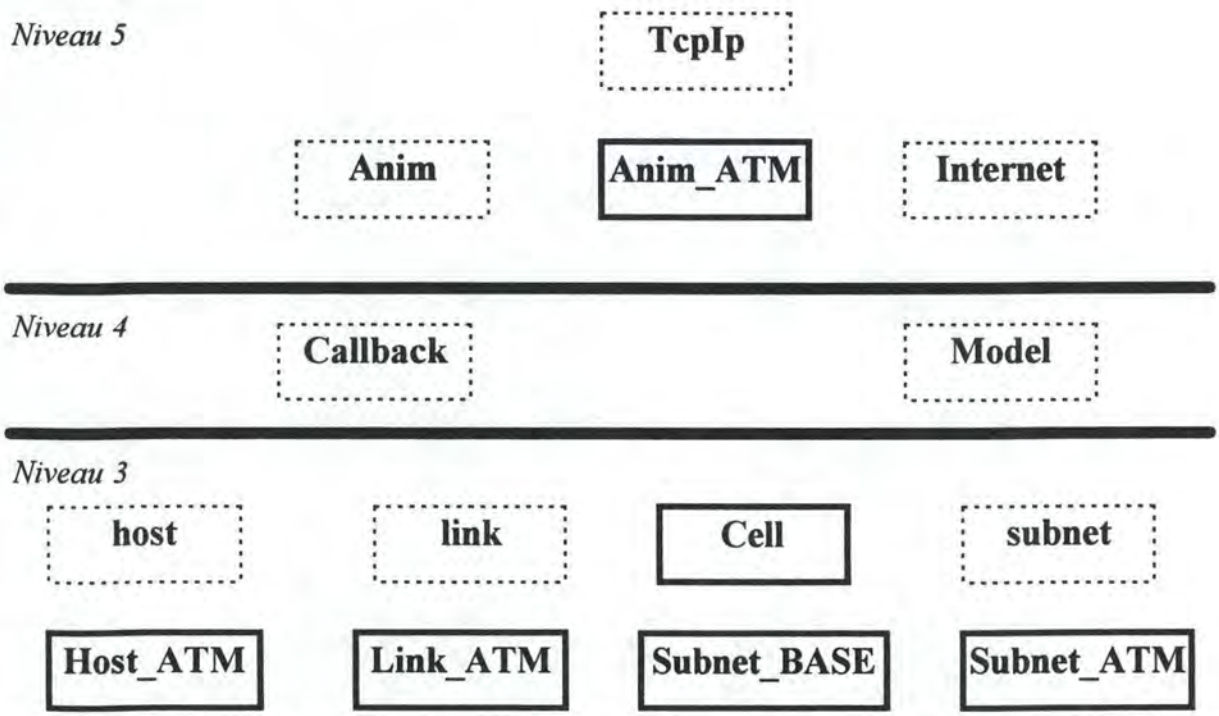


Figure 5-18 : La découpe en modules logiques du didacticiel

Le module de coordination "Anim_ATM" reprend toutes les fonctions relatives à la gestion de l'animation. Tous les autres modules introduits sont des modules de données et correspondent à la gestion des nouveaux objets. Le module subnet a été changé pour tenir compte des changements que nous avons apportés à la classe subnet. Les classes dérivées sont réparties dans deux modules : "Subnet_BASE" pour la gestion des objets correspondant à

[8] [XGO95] p.99

des réseaux déjà présents dans TcpIp et l'autre, "Subnet_ATM", pour la gestion des objets de type subnet_ATM.

Le module "Host_ATM" regroupe les fonctions de gestion et de traitement des objets qui sont une spécialisation de l'objet host (host_ATM, gate_ATM et switch_ATM). Le module "Link_ATM" intègre les fonctions de gestion et de traitement des objets dérivant de la classe link. Le module "Cell" s'occupe de tout ce qui concerne la classe cell.

Une fois la découpe en modules effectuée, il nous reste à décrire les différents modules et les services qu'ils offrent.

II. Le module Anim_ATM

Le module Anim_ATM a en charge toute la gestion de l'animation dans la fenêtre utilisée pour la vue détaillée des sous-réseaux lorsque le type de sous-réseau est ATM. Les fonctions que ce module prend en charge sont les suivantes.

- *Animate_ATM* : fonction qui lance l'animation en initialisant un timer qui a pour effet l'exécution de la fonction ATMRoutingTimerPROC une fois le temps associé à ce timer écoulé. Elle initialise également les variables qui serviront tout au long de l'animation.

- *ATMRoutingTimerPROC* : fonction qui représente le coeur de l'animation dans le sous-réseau ATM. Cette fonction est divisée en trois grandes parties : une pour chaque couche du protocole ATM. Chaque partie regroupe l'ensemble des étapes que nous avons dégagées lors de la décomposition en étapes établie au chapitre 4. Chaque exécution de la fonction provoque l'exécution d'une étape et la réinitialisation d'un timer qui entraînera une nouvelle exécution de cette fonction après un certain délai pour l'étape suivante.

- *ATMRoutingCellPROC* : fonction chargée de gérer la progression des cellules dans le réseau. Elle appelle indirectement la méthode traceInSubnet de l'objet subnetATM et initialise ensuite un timer dont le but est de provoquer une nouvelle exécution de cette fonction ATMRoutingCellPROC après un bref délai.

- *ResizeDatagramShell* : fonction qui adapte la taille de la fenêtre utilisée pour les PDU's en fonction de l'étape à laquelle on se trouve dans l'animation.

III. Le module Subnet_ATM

Le module Subnet_ATM centralise les définitions des méthodes de la classe subnetATM. Ce module inclut également des fonctions auxiliaires utilisées par les méthodes. Ces fonctions sont les suivantes :

- *Create_SubnetATM* : fonction d'interface entre le C et le C++. Cette fonction permet de créer un objet à partir de code C.

- *Create_link_list* : fonction qui crée la liste chaînée d'objets link liés à un objet switch_ATM.

- *Create_link* : fonction qui crée un objet de type link_ATM ou VP.

- *Create_name* : fonction qui crée le nom des machines à partir du nom générique du réseau auquel elles appartiennent et leur attribue une adresse IP.

- *WriteMemoText* : fonction qui affiche dans la fenêtre du sous-réseau, le nom de la couche pour laquelle l'animation se déroule.

- *createHostTable* : fonction qui remplit la table de routage (attribut table) de tous les objets d'un type dérivé de host et reliés à un objet subnet.

- *CreateWay* : fonction qui crée la liste des éléments du réseau s (argument) composant le chemin allant d'une machine à une autre (les références aux deux objets représentant ces machines sont passées en arguments).

- *clean_ATM_way* : fonction qui nettoie toute la mémoire réservée et réinitialise les variables intervenant dans l'animation du réseau ATM.

- *ReTraceHostAndLayer* : fonction qui retrace ce qui a été effacé dans les deux représentations des objets switch_ATM après que l'objet de type VP les reliant ait été tracé.

- *PrintSubStatusMessage* : fonction qui imprime dans la partie commentaire de la fenêtre du sous-réseau ATM, une phrase explicative correspondant au numéro passé en argument.

- *Which_Host_Is_In_way* : fonction qui détermine si une représentation d'un objet host_ATM ou gate_ATM est traversée par la représentation d'un objet VP.

- *Change_sens_of_links* : fonction qui intervertit l'origine et la destination de certains objets link_ATM. Chaque objet link_ATM relie deux switch_ATM dont un est qualifié d'origine et l'autre de destination. Si ces deux switch_ATM se retrouvent dans un des deux chemins ATM_way1 et ATM_way2 ou dans les deux et que leur ordre ne correspond pas à l'ordre (origine, destination) qui leur est attribué arbitrairement dans le link_ATM les reliant, cette fonction intervertit le pointeur vers l'objet origine et celui vers l'objet destination et elle gère les conséquences de ce changement.

- *Fill_ATM_routing_table* : fonction qui remplit la table de routage que l'on affiche dans la fenêtre prévue à cet effet.

IV. Le module Host_ATM

Le module Host_ATM rassemble les méthodes propres à la classe host_ATM ainsi que les fonctions utilisées par ces méthodes ou ayant rapport avec cette classe. Les fonctions sont les suivantes :

- *DrawLines* : fonction qui trace le contour des représentations des objets switch_ATM et dérivés ainsi que les arcs marquant la délimitation des couches .

- *DrawBackground* : fonction qui dessine la forme correspondant à la représentation d'un objet de type switch_ATM ou dérivé avec une couleur distinctive.

- *DrawAround* : fonction qui ne trace que le contour d'un objet switch_ATM ou dérivé avec un style de trait deux fois plus épais que la normale.

- *FillColor* : fonction qui dessine une couche d'une représentation d'un objet switch_ATM ou dérivé avec une couleur particulière (colorFill passée en argument).

- *HighLightLayer* : fonction qui dessine la couche "number" (Argument) d'un objet "host" (Argument) avec la couleur "fillColor"(Argument) . Cette fonction utilise les fonctions ci-dessus.

-
- *In_ATM_way* : fonction qui détermine si un objet de type switch_ATM fait partie d'une liste chaînée d'objets switch_ATM et renvoie sa position s'il est présent.

- *WhatIsAdress* : fonction qui renvoie un pointeur vers l'objet switch_ATM du réseau "sub" (Argument) dont l'adresse ATM est "atm_adr"(Argument).

- *WhichIsOnTop* : fonction qui détermine lequel des deux objets switch_ATM passés en argument a sa représentation plus haute dans la fenêtre que l'autre.

- *ReDrawText* : fonction qui réécrit le nom d'une couche dans la représentation d'un objet switch_ATM ou dérivé.

- *MAJroutingTable* : fonction qui met à jour les attributs multivalués d'un objet gate_ATM.

V. Le module Link_ATM

Le module Link_ATM est similaire au précédent. Il regroupe la définition des méthodes de la classe link_ATM et des fonctions se rapportant à cette classe. Ces fonctions sont les suivantes :

- *RetraceVP* : fonction d'interface entre le C et le C++ pour utiliser la méthode traceInSubnet

- *DeleteVP* : fonction d'interface entre le C et le C++ pour utiliser le destructeur de l'objet VP.

- *createOrDeleteVP* : fonction qui crée la connexion virtuelle VPath1 ou VPath2 selon la valeur du paramètre which. Si ces connexions existent, cette fonction les détruit.

VI. Le module Cell

Le module Cell regroupe les définitions des méthodes de la classe cell ainsi que les définitions des fonctions liées à cette classe. Les fonctions reprises dans ce module sont les suivantes :

- *CreateCell* : fonction d'interface entre le C et le C++ pour créer un objet cell.

- *DestructCell* : fonction d'interface entre le C et le C++ pour détruire un objet cell.

- *Create_or_delete_Cells* : fonction qui crée ou détruit selon la valeur de l'argument "create" autant d'objet cell qu'il y a d'objets link_ATM dans le réseau courant. Chaque objet cell est associé à un objet link_ATM.

- *ChangeValues* : fonction qui copie les attributs ATMway et color d'un objet cell vers un autre.

- *WhichCell* : fonction qui cherche quel est l'objet cell qui est associé à l'objet link_ATM passé en argument.

- *Copy_way* : fonction qui copie les coordonnées graphiques comprises dans l'attribut ATMway de la cellule VC_cell (objet global) vers une cellule passée en paramètre.

- *MoveCellInVC* : fonction qui s'occupe de gérer le déplacement des cellules sur la liaison virtuelle établie entre les deux couches AAL des machines sélectionnées par l'utilisateur.

- *MoveSecondCell* : fonction qui gère le déplacement des cellules entre les deux couches AAL des deux machines utilisées pour représenter une seconde connexion dans le réseau ATM.

- *MoveCellInPath* : fonction qui gère les déplacements des cellules dans la connexion virtuelle établie entre les deux couches ATM des deux machines sélectionnées par l'utilisateur.

- *MoveSecondCellInPath* : fonction qui gère les déplacements des cellules sur la deuxième connexion virtuelle créée pour illustrer la notion de multiplexage.

- *Move_in_PHY* : fonction qui gère le déplacement de toutes les cellules au niveau physique.

Chapitre 6

Outils logiciels de développement d'interfaces

Ce chapitre aborde différents outils d'aide à la conception d'interfaces. Depuis que les interfaces utilisateur existent, il y a des systèmes logiciels et des outils qui permettent d'aider la conception et l'implémentation de l'interface utilisateur. La plupart de ces outils ont fait leurs preuves en permettant des gains de productivité significatifs pour le programmeur. D'autres ont rencontré moins de succès quant aux possibilités d'aide à la construction d'interfaces telles que les utilisateurs les souhaitent.

Nous allons présenter différents types d'outils existant avec leurs avantages et inconvénients dans l'optique de développer notre interface telle que nous l'avons décrite dans le chapitre précédent. Nous nous sommes basés sur l'article de Brad A. Myers [Mye94] pour classer les outils qui vont être décrits.

6.1. Introduction

La création d'une interface utilisateur est souvent une tâche complexe et qui occupe une place importante dans le logiciel. L'interface que l'on souhaite réaliser est souvent difficile à implémenter, à débuger et à modifier. Une étude à ce sujet révéla qu'en moyenne 48% du code d'une application était dédié à l'interface et que 50% du temps était consacré à l'implémentation de la partie interface. Notre logiciel d'animation graphique se situe largement au-dessus de la moyenne en terme de pourcentage de lignes de code et de temps consacrés à l'interface.

Plus les interfaces deviennent faciles à utiliser, plus il devient difficile de les créer. Aujourd'hui, les interfaces à manipulation directe, encore appelées GUI (*Graphical User Interface*) sont pratiquement répandues partout. Une étude de 1993 montra que 97% de tous les logiciels développés sous UNIX possédaient une GUI. Par conséquent, puisque la conception d'interfaces utilisateur est si difficile, la seule méthode pour obtenir de bonnes interfaces est de procéder itérativement. Ces itérations reprennent les phases de conception et d'implémentation

après chaque test auprès de l'utilisateur. Mais cette méthode rend l'implémentation encore plus difficile.

Heureusement, de sérieux progrès ont été faits en matière de logiciel d'aide à la création d'interfaces si bien que maintenant, pratiquement tout logiciel comportant une interface utilisateur est créé à l'aide d'outils facilitant l'implémentation. Par exemple, le système MacApp d'Apple a permis de réduire le temps de développement d'un facteur 4 ou 5 dans certains cas. Un autre exemple affirme qu'un programme d'application moyen développé avec l'environnement NeXTStep permet une économie de 83% de lignes de code et de 50% de temps comparé à une application écrite en utilisant des outils moins avancés.

6.2. Importance des outils de développement d'interface

Il y a plusieurs avantages à utiliser des outils d'aide au développement d'interfaces utilisateur. Ces avantages peuvent être classés en deux groupes principaux.

1 La qualité de l'interface sera meilleure parce que :
- 1.1 Les modèles peuvent être rapidement prototypés et implémentés.
- 1.2 Les changements consécutifs aux tests sont faciles à incorporer.
- 1.3 Des interfaces multiples peuvent exister pour une même application.
- 1.4 Les applications auront des interfaces plus cohérentes entre elles si le même outil est utilisé.
- 1.5 Les spécialistes de différents domaines impliqués dans la conception de l'interface seront plus nombreux.

2 Le code relatif à l'interface sera créé et maintenu de manière plus facile et plus économique parce que :
- 2.1 Les spécifications peuvent être représentées, validées et évaluées plus facilement.
- 2.2 Le code à écrire est réduit.
- 2.3 La découpe en modules sera meilleure à cause de la séparation entre l'interface et l'application elle-même. Cela permettra de changer l'un ou l'autre module sans toucher au reste de l'application.
- 2.4 Le niveau d'expertise des concepteurs d'interfaces peut être moindre parce que l'outil cache une grande partie de la complexité sous-jacente du système.
- 2.5 La fiabilité de l'interface sera plus grande à cause de la génération automatique du code à partir de spécifications de plus haut niveau.
- 2.6 Les logiciels seront plus facilement portables si l'interface peut être re-générée sur différents systèmes supportant le même outil.

En se basant sur ces buts, nous pouvons énumérer un certain nombre de fonctions qui devraient être fournies par les outils d'aide à la conception d'interface. Cette liste peut être utilisée pour évaluer les différents outils.

En général, l'outil devrait pouvoir :

- Aider à la conception de l'interface étant donné une spécification de la tâche de l'utilisateur.
- Aider à l'implémentation étant donné une spécification du modèle de l'interface.

- Aider à l'évaluation de l'interface et proposer des améliorations ou fournir des informations pour aider le concepteur à évaluer l'interface.
- Créer des interfaces faciles à utiliser.
- Permettre à des non-programmeurs de concevoir et implémenter l'interface.
- Permettre à l'utilisateur de personnaliser son interface.
- Autoriser la portabilité.
- Etre facile à utiliser.

Un outil de développement d'interfaces peut être divisé en différentes couches : le système de fenêtrage, le toolkit et les outils de plus haut niveau. La *Figure 6-1* illustre les composants d'un outil d'aide à la conception d'interfaces.

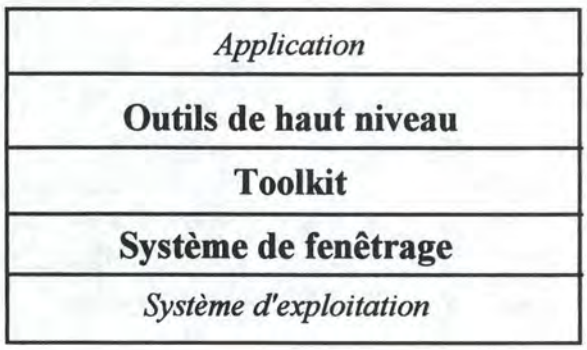


Figure 6-1 : Composants d'un outil d'aide à la conception d'interfaces

Le système de fenêtrage (*Windowing System*) s'occupe de la gestion des fenêtres. Le toolkit que l'on peut traduire par "boite à outils" contient les objets de base comme les boutons, les menus, les barres défilantes et les champs éditables. Au-dessus de ce toolkit peuvent se trouver les outils de plus haut niveau (*Higher Level Tools*) qui aident le concepteur à utiliser les objets du toolkit.

6.3. Système de fenêtrage

Un système de gestion de fenêtres est un logiciel qui aide l'utilisateur à gérer et contrôler différents contextes en les séparant physiquement en différentes régions de l'écran appelées fenêtres. Dans les années 70, début 80, il existait un grand nombre de systèmes de gestion de fenêtre différents. Chacun de ceux-ci ne tournait que sur sa propre configuration matérielle. Les programmeurs devaient donc, s'ils voulaient faire tourner leur logiciel sur différentes plates-formes, réécrire un grand nombre de lignes de code.

Le système de gestion de fenêtrage X windows fut créé pour résoudre ce problème en fournissant une interface indépendante du matériel. X fut un succès et a très vite surclassé tous les autres systèmes dans le monde des stations de travail. C'est d'ailleurs ce système que nous avons utilisé pour notre logiciel. Dans le marché des petits ordinateurs, Macintosh a son propre système de fenêtrage ou utilise X et les PC utilisent principalement Microsoft Windows ou le Presentation Manager d'IBM.

6.4. Toolkits

Un toolkit est une librairie d'objets (*Widgets*) qui peuvent être appelés par un programme d'application. La création d'une interface en utilisant un toolkit ne peut être réalisée que par un programmeur expérimenté. En effet, les toolkits n'ont qu'une interface procédurale qui implémente les fonctions de base du système de fenêtrage.

L'avantage d'utiliser un toolkit est que l'interface ainsi obtenue ressemblera et se comportera de la même manière que les autres interfaces créées avec le même toolkit. Le problème est que les styles d'interaction sont limités à ceux fournis. Le second problème des toolkits est qu'ils sont souvent difficiles à utiliser parce qu'ils contiennent des centaines de procédures et la manière de les utiliser pour obtenir l'interface désirée n'est pas toujours claire.

Avec X windows, les programmeurs peuvent utiliser plusieurs toolkits dont par exemple Xt que nous avons utilisé pour notre logiciel, Interviews, Garnet ou tk.

Comme les concepteurs de X n'ont pas pu se mettre d'accord sur une présentation unique (*look-and-feel*) de ces objets, ils ont créé une couche intrinsics sur laquelle on construirait les ensembles d'objets présentant des aspects et des comportements différents. Cette nouvelle couche fournit des services tels que des techniques de programmation Orientée Objet et de contrôle de présentation.

Plusieurs ensembles d'objets présentant des aspects et des comportements différents peuvent être implémentés au-dessus de différentes couches intrinsics (*Figure 6-2(a)*) ou bien, le même ensemble peut être implémenté au-dessus de différentes couches intrinsics (*Figure 6-2(b)*).

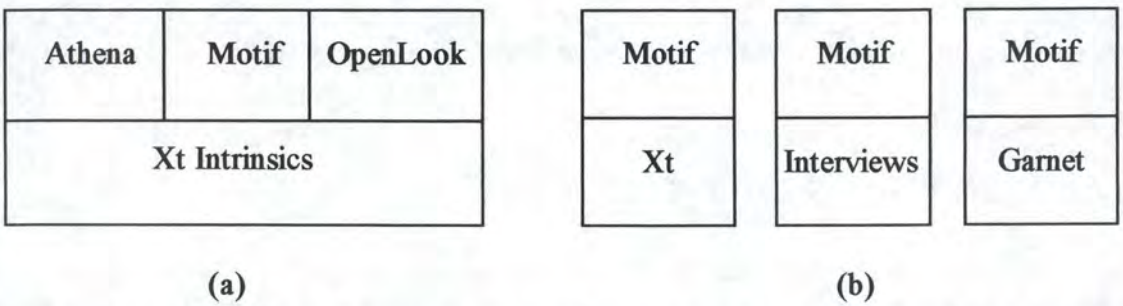


Figure 6-2 : (a) différents ensembles d'objets créés sur la même couche intrinsics (b) le même ensemble d'objets créé sur trois intrinscs différents

6.5. Outils de plus haut niveau

La programmation au niveau des toolkits est assez difficile d'où l'énorme intérêt porté aux outils de plus haut niveau pour faciliter grandement le processus de production de l'interface utilisateur. Nous n'avons utilisé aucun des outils qui vont être présentés ci-dessous. Cependant, pour la création de notre interface, nous avons utilisé UI2C. Il s'agit d'un outil développé au CEGELY par Daniel Muller et spécifié dans [MUL94]. Nous en reparlerons lorsqu'on présentera les classes d'outils.

6.5.1. Phases

Les outils basés sur les toolkits possèdent plusieurs composants qui interviennent à différents moments. Le composant d'aide à la conception assiste le concepteur dans la création de son interface. Par exemple, il peut s'agir d'un éditeur graphique qui aide à la disposition spatiale des objets. La phase suivante concerne l'utilisation de l'interface par l'utilisateur final. Ici, c'est le composant lié à l'exécution qui est utilisé. Ce composant gère l'interface.

Il peut aussi y avoir des composants utilisables après l'exécution pour aider à l'évaluation et au débugage de l'interface. Malheureusement, très peu d'outils disposent actuellement de ce dernier composant. Une nouvelle génération d'outils essayent d'évaluer la manière dont les gens interagissent avec l'interface en créant automatiquement des modèles cognitifs à partir de descriptions de haut niveau de l'interface utilisateur.

6.5.2. Styles d'interfaces

Les outils d'aide à la conception d'interface présentent différentes formes. Un critère important de classement est la manière dont les concepteurs spécifient l'interface. Certains outils exigent que le programmeur utilise un langage spécifique ; d'autres fournissent une structure d'application pour guider la programmation ; d'autres encore génèrent automatiquement l'interface à partir d'un modèle ou de spécifications de haut niveau et d'autres enfin permettent la conception interactive de l'interface. Passons en revue ces différents outils avec leurs avantages et inconvénients.

6.5.3. Outils basés sur le langage

A l'aide de la plupart des outils les plus vieux, les concepteurs spécifient leurs interfaces dans un langage propre. Ces langages peuvent prendre plusieurs formes telles que les grammaires hors contexte, les diagrammes d'états-transitions, les langages descriptifs, les langages événementiels, etc. Le langage est utilisé d'habitude pour spécifier la syntaxe de l'interface, c'est-à-dire la séquence des actions d'entrées/sorties encore appelées dialogues.

Un des problèmes de ces langages est qu'ils ne peuvent être utilisés que par des programmeurs professionnels. De plus, il semble plus naturel de définir la partie graphique d'une interface en utilisant un éditeur graphique. Cependant, à l'avenir, la plupart des interfaces auront encore besoin d'être créées par programmation. Ces types d'outils n'auraient évidemment pas pu nous aider à concevoir notre interface hautement graphique.

6.5.4. Outils fournissant un canevas d'applications

Après l'apparition du toolkit de Macintosh, Apple s'est rendu compte des difficultés qu'avaient les programmeurs à savoir quand utiliser les différentes fonctions fournies. Apple a alors créé un système qui fournit aux programmeurs une structure globale d'application pour les guider. Ce système appelé MacApp fournit des classes d'objets prédéfinies pour les parties importantes d'une application comme la fenêtre principale et les commandes. Il reste aux programmeurs à spécialiser ces classes en fournissant les détails spécifiques comme ce qui doit être dessiné dans la fenêtre principale ou quelles sont les commandes de leur application.

Le système ACE de HP fournit un éditeur interactif qui permet de spécifier quelques propriétés des objets mais la plupart des comportements spécifiques à l'application doivent encore être programmés.

Toutes ces structures exigent quand même un effort important d'écriture de code de la part du concepteur.

Pour notre application, comme il s'agissait de greffer une partie nouvelle sur une structure existante, ce genre d'outils ne nous aurait servi à rien.

6.5.5. Outils de génération automatique de code

Lorsque le concepteur utilise des outils basés sur le langage, il doit spécifier en grande partie les caractéristiques des objets telles que leur format et leur placement. En utilisant les outils de génération automatique, le choix des caractéristiques des objets n'incombe plus au concepteur mais sont pris en charge par l'outil. Pour faire ses choix, l'outil se base sur des spécifications de plus haut niveau. La plupart de ces outils se sont concentrés sur la création de menus et de boites de dialogues.

Les deux problèmes majeurs liés à ce type d'outils sont d'une part que les interfaces générées ne sont généralement pas assez bonnes, même si certains outils permettent de les éditer une fois générées, et d'autre part que les langages de spécification restent complexes à utiliser et à apprendre. Il faudrait également élargir la gamme d'objets qui peuvent être générés automatiquement par ces outils.

Le seul outil que nous avons utilisé pour concevoir notre interface plus facilement appartient à cette classe. En effet, UI2C dont nous avons déjà parlé, permet de décrire d'une manière simple, les propriétés et les comportements événementiels des objets (*Widgets*) formant les fenêtres de notre interface. UI2C génère ensuite du code C comprenant des fonctions fournies par le toolkit Xt.

6.5.6. Outils de spécification graphique

Les outils décrits dans cette section permettent, du moins partiellement, de définir l'interface en plaçant les objets à l'écran par manipulation directe. L'idée à la base de ces outils part de l'observation que la présentation visuelle de l'interface est ce qu'il y a de plus important dans la création d'une interface graphique. Ces outils ont l'avantage supplémentaire d'être faciles à utiliser.

Les outils permettant une telle spécification graphique peuvent être classés en quatre catégories : les outils de prototypage, ceux qui se basent sur la métaphore des cartes, les constructeurs d'interfaces et les éditeurs pour des graphiques spécifiques à une application.

I. Outils de prototypage

Le but des outils de prototypage est de permettre au concepteur d'établir rapidement une maquette des écrans de son programme. Le gros problème de ces outils est qu'ils permettent uniquement de créer une image des écrans sans en générer par la suite le code pour les programmer.

Pour les interfaces graphiques, les concepteurs utilisent souvent des outils tels que Director pour Macintosh qui est réellement un outil d'animation. Avec ces outils, le concepteur peut dessiner des écrans et spécifier que, lorsque tel événement se produit, une animation doit démarrer ou un autre écran doit apparaître. Néanmoins, les capacités de visualiser les comportements demeurent limitées. HyperCard pour Macintosh est aussi souvent utilisé comme outil de prototypage bien qu'il s'agisse avant tout d'un outil basé sur la métaphore des cartes (voir point suivant).

Si de tels outils avaient été disponibles dans l'environnement sur lequel nous travaillions, nous aurions pu les utiliser pour créer une maquette de notre interface. Il semble que ces outils auraient été les plus aptes à nous aider dans la conception de notre interface parmi tous les outils présentés dans ce chapitre. Cependant, compte tenu de leur désavantage majeur en terme de génération de code, le gain de temps potentiel lié à leur utilisation aurait été négligeable d'autant plus qu'il aurait fallu apprendre à les utiliser endéans un court laps de temps.

II. Outils basés sur la métaphore des cartes

Beaucoup de programmes graphiques ont des interfaces limitées à une séquence de pages relativement statiques. C'est le cas par exemple de didacticiels utilisés dans le domaine de l'enseignement aux handicapés. Dans ces programmes, chaque page contient un ensemble d'objets dont certains servent à provoquer le passage d'une page à l'autre.

L'exemple probablement le plus connu de systèmes basés sur le système de cartes est HyperCard d'Apple. Avec cet outil, le concepteur peut facilement créer des cartes dans lesquelles il place des champs, des images, des sons, des boutons, etc. Cet outil fournit également un langage de rédaction de scripts pour assigner davantage de flexibilité aux boutons que le simple passage d'une page à l'autre.

Nous aurions pu utiliser ce genre d'outils mais il est probable que le nombre d'objets à gérer lors de l'animation aurait rendu les performances du système trop mauvaises.

III. Outils "constructeurs d'interfaces"

Un outil qualifié de constructeur d'interfaces (*interface builder*) permet au concepteur de créer des boites de dialogues, des menus et des fenêtres. Le concepteur sélectionne les objets qu'il désire parmi les librairies fournies et les place à l'écran en utilisant la souris. Les autres propriétés des objets, outre la place à l'écran, peuvent être définies en remplissant des feuilles de spécification de propriétés.

La plupart de ces outils génèrent des modèles (*template*) en C qui peuvent être compilés avec le code de l'application. WindowsMAKER pour Microsoft Windows sur PC fonctionne de cette manière. D'autres génèrent une description de l'interface dans un langage qui peut être lu à l'exécution. C'est le cas par exemple de UIMX pour X Windows et Motif qui génère une description en UIL (*User Interface Language*).

Ce genre d'outils compte comme désavantage de ne fournir que très peu d'assistance au concepteur dans sa tâche en lui laissant une grande liberté dans le placement des objets notamment. Un autre problème est à relever pour les interfaces graphiques. Ils ne permettent pas de dessiner de telles interfaces. Pour cette raison, ce genre d'outils ne nous aurait pas convenu.

IV. Outils "éditeurs pour graphiques spécifiques"

Quand une application est composée de dessins particuliers comme c'est le cas pour notre interface, il serait utile pour le concepteur de pouvoir dessiner ces graphiques plutôt que de devoir les programmer. Le problème avec ce genre d'objets graphiques est qu'ils changent souvent lors de l'exécution. Dès lors, le concepteur ne peut dessiner qu'un exemple de ce que l'interface désirée serait. Des outils permettant de transformer ces exemples en prototypes sont encore au stade de la recherche.

6.6. Conclusion

Il existe un large éventail d'outils répondant à une multitude d'approches. En outre, plusieurs approches peuvent être appropriées pour réaliser une même tâche, c'est pourquoi il est utile de faire une évaluation de tous les outils. Une chose importante est à noter à l'heure actuelle : pour pratiquement toutes les tâches, il est probable qu'il existe un outil de haut niveau. Donc, si on programme directement au niveau du système de fenêtrage ou au niveau du toolkit, il doit exister un outil pour nous faire gagner du temps.

Malgré ces constatations, nous avons programmé au niveau des toolkits en utilisant Motif et Xt avec X Windows. Les raisons ont déjà été évoquées à plusieurs reprises mais nous allons les rappeler :

- la spécificité graphique de notre application aurait difficilement pu être supportée par un outil du marché ;
- si un tel outil existait sur le marché, il ne se trouvait pas à Lyon dans l'environnement sur lequel nous avons travaillé ;
- le contrôle que permet la programmation de "bas niveau" ne se retrouvera jamais si l'on passe par un outil intermédiaire non spécialisé pour le genre d'applications que nous voulons développer et
- le but de notre logiciel était aussi d'apprendre à programmer à bas niveau pour comprendre ce style de programmation et pouvoir tirer un meilleur profit et utiliser des outils d'aide à la conception pour des applications futures.

Conclusion

La tâche que représente la maintenance évolutive d'un logiciel quelconque n'est jamais évidente. Notre travail s'est porté sur l'ajout d'une extension au didacticiel TcpIp. Cette nouvelle partie présente le protocole ATM sous forme de simulation. Pour cela, il a fallu maîtriser l'architecture du logiciel existant, les notions propres au protocole ATM et tenir compte de notions pédagogiques, ergonomiques et informatiques.

Le travail que nous avons effectué ne s'est pas du tout attaché à combler certaines lacunes de didacticiel comme, par exemple, l'absence d'aide en ligne ou l'absence de fonction de contrôle vis-à-vis de l'atteinte des objectifs par l'élève. Dans la partie existante, nous avons seulement apporté des changements dans l'architecture logicielle pour la rendre plus facilement évolutive.

La partie que nous avons ajoutée et qui simule le protocole ATM est relativement indépendante par rapport au reste du logiciel. Nous sommes partis de l'idée que l'élève ne sait pas porter son attention, à la fois sur l'apprentissage des notions propres à ATM, et sur ce qui se passe au niveau supérieur, c'est-à-dire au niveau de l'Internet. Comme notre priorité était plutôt la partie ATM seule, l'intégration du sous-réseau ATM dans l'Internet est minimale et reste à améliorer.

Nous regrettons de n'avoir pas eu assez de temps pour réaliser cette intégration de manière plus optimale. De plus, il ne nous a pas été possible de réaliser suffisamment de tests d'ensemble sur la partie ATM par manque de temps. Cependant, chaque fonctionnalité de la partie ATM a été testée individuellement puisque nous avons suivi une approche de développement par prototypage.

Un des défauts majeurs qui mériterait d'être corrigé est la taille des fenêtres utilisées pour la simulation de la partie ATM. Nous avons utilisé de grandes fenêtres dans un souci du détail mais ce qui pourrait apparaître comme un avantage au niveau des grands écrans (21"), se révèle être un problème lors de l'utilisation d'écrans plus petits (17").

La continuation du projet pourrait être faite en utilisant certains outils parmi ceux que nous avons présentés au chapitre 6. De plus, les points faibles de la partie initiale de TcpIp et d'autres lacunes qui viennent d'être citées sur la partie ATM pourront être améliorées.

Et enfin, nous pouvons dire que le test réalisé auprès de certains étudiants de l'Institut d'Informatique s'est révélé très concluant pour certaines fonctionnalités du logiciel comme le

passage des PDU's entre la couche IP les couches du protocole ATM. Cependant, de nombreuses suggestions ont été faites pour résoudre les ambiguïtés et autres difficultés subsistantes. Ces points, pouvant être améliorés, sont repris ci-dessous.

pour la partie Internet.

- Mettre une légende dans l'Internet pour expliquer quels sont les types de réseaux.
- Créer le moyen de paramétrer l'interface de l'Internet.
- Changer certains termes inappropriés dans la fenêtre de choix d'une passerelle.
- Laisser en couleur les passerelles par lesquelles le datagramme IP passe lors de la simulation entre les deux machines sélectionnées par l'utilisateur.
- Déplacer plus à droite la représentation du réseau vertical de droite dans l'Internet pour que les lignes louées soient plus visibles.
- Changer la couleur du sous-réseau ATM dans l'Internet (même couleur que dans la fenêtre du sous-réseau) lorsqu'il est vu de manière détaillée dans la fenêtre des sous-réseaux.

pour la partie ATM.

- Représenter la signalisation de manière simplifiée au niveau AAL mais pas aux niveaux inférieurs pour ne pas surcharger la représentation existante. Il faudra évidemment mentionner pourquoi la signalisation n'est pas représentée aux niveaux inférieurs.
- Mettre plus en évidence la caractère asynchrone aux niveaux ATM et physique
- Expliquer la raison de la deuxième connexion.
- Accentuer la différence de couleurs entre les deux connexions.
- Augmenter le nombre de cellules émises sur la connexion principale.
- Changer la manière dont les champs vides dans les PDU's sont représentés dans la fenêtre des PDU's.
- Changer la représentation du format des cellules physiques qui ne sont que des bits sans format (la raison de laisser le format a été expliquée au chapitre 4 §1.5 III).

Abréviations

AAL	ATM Adaptation Layer
AAL-PDU	AAL Protocol Data Unit
AAL-SDU	AAL Service Data Unit
AAPL	APPLication layer
ATM	Asynchronous Transfer Mode
ATM-PDU	ATM Protocol Data Unit
ATM-SDU	ATM Service Data Unit
BASize	Buffer size
Btag	Beginning tag
CCITT	Consultative Committee for International Telegraph and Telephone
CEGELY	CEntre de GEnie électrique de LYon
CLP	Cell Loss Priority
COLOS	COnceptual Learning Of Science
CONS	Connection Oriented Network Service
CPCS	Common Part Convergence Sublayer
CPCS-PDU	CPCS Protocol Data Unit
CPI	Common Part Identifier
CRC	Cyclic Redundancy Check
CS	Convergence Sublayer
CS-PDU	CS Protocol Data Unit
CSPDN	Circuit Switched Protocol Data Network
DARPA	Defense Advanced Research Project Agency
DOD	Department Of Defense
EAO	Enseignement Assisté par Ordinateur
Etag	Ending tag
FTP	File Transfer Protocol
GC	Graphic Control
GFC	Genereric Flow Control
GUI	Graphical User Interface
HEC	Header Error Control
HP	Hewlett Packard
HTTP	HyperText Transport Protocol
IP	Internet Protocol
ISO	International Standard Organisation
ITU-T	International Telecommunication Union
LAN	Local Area Network.
LI	Length Indicator

MID	Multiplexing IDentifier
N-PCI	Protocol Control Information of layer N
N-PDU	Protocol Data Unit of layer N
N-SDU	Service Data Unit of layer N
NNI	Network to Network Interface
OAM	Operation And Maintenance
OSI	Open System Interconnection
PAD	PADding
PC	Personal Computer
PCI	Protocol Control Information
PDU	Protocol Data Unit
PM	Physical Media
PSPDN	Packet Switched Public Data Network
PT	Payload Type
RNIS	Réseau Numérique à Intégration de Service
SAP	Service Access Point
SAR	Segmentation And Reassembly
SAR-PDU	SAR Protocol Data Unit
SAR-SDU	SAR Service Data Unit
SDH	Synchronous Digital Hierarchy
SDU	Service Data Unit
SN	Sequence Number
SNP	Sequence Number Protection
SSCS	Service-Specific Convergence Sublayer
ST	Segment Type
STM	Synchronous Transfer Mode
TC	Transmission Convergence
TCP	Transmission Control Protocol
UIL	User Interface Language
UNI	User to Network Identifier
URL	Uniform Resource Locator
UU	CPCS User-to-User indication
VC	Virtual Channel
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VP	Virtual Path
VPC	Virtual Path Connection
VPI	Virtual Path Identifier

Bibliographie

[COM91] D. E. COMER, INTERNETWORKING WITH TCP/IP. Vol 1 - Principles, Protocols and Architecture, Second Edition, Prentice Hall International, Inc., USA, 1991.

[CRO90] Kel CROSSLEY et Les GREEN, LE DESIGN DES DIDACTICIELS, Art Culture Lecture - Editions, Paris, 1990.

[DUB94] Eric DUBOIS, Méthodologie de Développement de Logiciel (MDL). Cours de seconde licence et maîtrise en informatique, Institut d'Informatique, Facultés Universitaires Not&re-Dame de la Paix, Namur, 1994.

[HAN95] R. HÄNDEL, M. N. HUBER, S. SCHRÖDER, COMPRENDRE ATM, Addison-Wesley, Paris, 1995.

[HPC89] HP, HP OSF/MOTIF PROGRAMMER'S REFERENCE. HP 9000 Series 300/800 Computers, Hewlett-Packard Company, USA.

[JAV95] URL : http://java.sun.com/doc/Overviews/java/index.html.

[JVD95] Jean VANDERDONCKT, Interface Homme-Machine, Cours de seconde licence et maîtrise en informatique, Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix , Namur, 1995.

[KER88] Brian W. KERNIGHAN, Dennis M. RITCHIE, THE C PROGRAMMING LANGUAGE, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[KYA95] Othmar KYAS, ATM NETWORKS, International Thomson Publishing, Sheffield, 1995.

[MEY92] Scott MEYERS, EFFECTIVE C++ - 50 specific ways to improve your programs and designs, Addison Wesley, Reading, Massachusetts, 1992.

[MUL94] Daniel MULLER, "UI2C : a User Interface to C language translator", CEGELY, Ecole Centrale de Lyon, Lyon, 1994.

[MUL94a] Daniel MULLER, "The Standard COLOS platform", 3rd Edition, CEGELY, Ecole Centrale de Lyon, Lyon, 1994.

[MUL95] Daniel MULLER, "The Colos Project", URL : http://www.colos.ec-lyon.fr/colosHp/colosEurope/, CEGELY, Ecole Centrale de Lyon, Lyon.

[MYE94] Brad A. MYERS, "User Interface Software Tools", Submitted for publication, 8 août 94. This is a revised version from : Brad A. MYER, "State of the Art in User Interface Software Tools", Advances in Human Computer Interaction, Volume 4, Edited by H. Rex Hartson and Deborah Hix, Norwood, NJ, 1993, pp. 110-150.

[PRY91] Martin DE PRYCKER, ASYNCHRONOUS TRANSFER MODE, Solution for Broadband ISDN, Ellis Horwood, London, 1991.

[STR95] Bjarne STOUSTRUP, THE C++ PROGRAMMING LANGUAGE, Second Edition, Addison Wesley, Reading, Massachusetts, 1995.

[VBA93] Philippe VAN BASTELAER, Véronique NACHTERGAELE, Notes provisoires pour les cours de Téléinformatique et réseaux, Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix, Namur, 1994.

[XGO95] Xavier GOBERT, Contribution à l'Enseignement Assisté par Ordinateur dans le domaine des télécommunications. Un didacticiel pour Internet, mémoire présenté en vue de l'obtention du grade de licencié et maître en informatique, Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix, Namur, 1995.

[YOU94] Douglas A. YOUNG, THE X WINDOW SYSTEM - PROGRAMMING AND APPLICATIONS WITH XT, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1994.